

تعلــم

ماتلاب MATLAB بنفسك

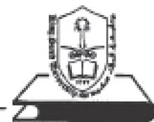
خطوة بخطوة تعلم بنفسك برنامج ماتلاب
من خلال العديد من الأمثلة العامة والمتخصصة

تأليف

أ.د. محمد إبراهيم العدوي
أستاذ هندسة الاتصالات – جامعة حلوان مصر

أ.د. حسن فؤاد محمد السيد
أستاذ الهندسة الطبية – جامعة الملك سعود

د. نانسي مصطفى سالم
مدرس بقسم الهندسة الطبية كلية الهندسة
جامعة حلوان – مصر



النشر العلمي والمطابع - جامعة الملك سعود

ص.ب ٦٨٩٥٣ - الرياض ١١٥٣٧ - المملكة العربية السعودية

فهرسة مكتبة الملك فهد الوطنية أثناء النشر

العدوي، محمد ابراهيم.

تعلم ماتلاب MATLAB بنفسك / محمد إبراهيم العدوي ؛ حسن فؤاد محمد السيد ؛ نانسي مصطفى سالم – الرياض، ١٤٣٤هـ.

٣٠١ ص ؛ ١٧سم × ٢٤ سم

ردمك: ٢ - ٠٨١ - ٥٠٧ - ٦٠٣ - ٩٧٨

١ – لغة ماتلاب ٢– لغات البرمجة (حواسيب) ٣– الرياضيات –معالجة البيانات أ. السيد، حسن فؤاد محمد (مؤلف مشارك) ب. سالم، نانسي مصطفى (مؤلف مشارك) ج. العنوان

ديوي ٠٠٥,١٣٣ ١٤٣٤/٢١٨٠

رقم الإيداع: ١٤٣٤/٢١٨٠

ردمك: ٢ - ٠٨١ - ٥٠٧ - ٦٠٣ - ٩٧٨

حكمت هذا الكتاب لجنة متخصصة، وقد وافق المجلس العلمي على نشره في اجتماعه الرابع للعام الدراسي ١٤٣٤/١٤٣٣هـ، المعقود بتاريخ ١٤٣٣/١١/٢٨هـ، الموافق ٢٠١٢/١٠/١٤م.

الإدارة العامة للنشر العلمي والمطابع تعتذر عن عدم وضوح بعض الأشكال لعدم تمكن المؤلف من إحضار أشكال بجودة مناسبة للطباعة



النشر العلمي والمطابع ١٤٣٤هـ

المقدمة

يوجد في دنيا الحاسبات العديد من البرمجيات التطبيقية ، والتي من أكثرها شيوعا مجموعة برمجيات Microsoft office والتي تحتوي على العديد من البرمجيات التى يستخدمها يوميا كل من يتعامل مع الحاسب وفي أي تخصص. ونحن نعتقد أن برنامج ماتلاب MATLAB من شركة ماثورك Mathwork يأتي في المرتبة التالية من حيث الشيوع والاستخدام بعد برمجيات Microsoft office وبالذات بين المتخصصين في المجالات العلمية ، والتي من أشهرها الهندسة (بجميع تخصصاتها) والرياضيات.

القليل جدا من الجامعات والمعاهد العلمية يقوم بتدريس ماتلاب كمادة منفصلة بمحتوى علمي محدد في أي سنة من سنواتها الدراسية ، ولكن في الغالب مايتم استخدامه كبرنامج تطبيقى يتم الإشارة إليه عند الحاجة لحل تمرين أو إجراء مشروع معين في أحد المقررات الدراسية مثل مقرر معالجة الإشارات الرقمية ، أو مقرر الأنظمة والإشارات ، أو أحد مقررات الرياضيات ، أو الميكانيكا أو التحكم الآلي. في كل هذه المقررات يحتاج الطالب لحل الكثير من التمارين أو إجراء بعض المشروعات مستخدما الماتلاب دون أن يكون لديه فكرة عن هذا البرنامج من قبل ؛ لذلك فإن الطالب في هذه

الحالة يكون في عجلة من أمره لتعلم هذا البرنامج. وبالطبع ، فإن هذا الوقت يكون مقتطعا من وقت المقرر الذى يدرسه.

لذلك ؛ فقد كنا حريصين في هذا الكتاب على أن نقدم برنامج ماتلاب في صورة مبسطة وسهلة بحيث يمكن للطالب أن يعتمد على نفسه كليا دون الحاجة إلى اللجوء لأي مكتب من المكاتب التي تقوم بتدريس هذا المقرر مجزءا على عدة دورات بتكلفة عالية جدا ؛ ولذلك فإننا ننصح أي طالب في السنوات الأولى من دراسته الجامعية أن يقوم باقتناء هذا الكتاب ؛ ويبدأ في تعلم برنامج ماتلاب معتمدا على نفسه وخاصة طلاب كليات علوم الحاسب و الهندسة (بجميع تخصصاتها).

إن برنامج ماتلاب من الصعب جدا أن يتم جمعه أو تقديمه كاملا في كتاب واحد حيث أنه يستخدم في جميع التخصصات بلا استثناء وبخاصة الهندسية منها. ولذلك فقد راعينا في هذا الكتاب أن يقدم المحتويات العامة المستخدمة من قبل جميع التخصصات ، مثل استخدام الماتلاب في حالته التفاعلية مع المستخدم ؛ والتي من خلالها يمكن للمستخدم حل الكثير من المشاكل أو التمارين بسرعة كبيرة وبصورة تفاعلية كما لو كان يستخدم الآلة الحاسبة تماما. الصورة الثانية من صور استخدام ماتلاب هي صورته البرمجية التي يستخدم فيها كأي لغة برمجة عادية من لغات البرمجة ذات المستوى العالي مثل لغة C^{++} ، ولقد تم تقديم كل من هاتين الصورتين في هذا الكتاب بالتفصيل. الصورة الثالثة من صور استخدام برنامج ماتلاب هي صورة المحاكاة ، حيث يمكن استخدام ماتلاب في محاكاة أي مشكلة أو أي مشروع ، ثم إدخال إشارات الدخل لهذا المشروع وتجميع إشارات الخرج ورؤيتها على العديد من أجهزة عرض الإشارات التي يمكن سحبها وإسقاطها على مساحة العمل في برنامج المحاكاة ورؤية المشروع الذي تمت محاكاته وهو يعمل بصورة كاملة وتحت أي ظرف من ظروف العمل.

يحتوي هذا الكتاب على أحد عشر فصلا موزعة كالتالي :

الفصل الأول : عبارة عن مقدمة عامة عن برنامج الماتلاب تم وضعها في صورة سهلة ومباشرة ، والغرض منها فقط هو استعراض مقدرة وإمكانات برنامج ماتلاب المختلفة من خلال مجموعة برامج يمكن تنفيذها بسهولة والحصول على نتائجها ، وأما التفاصيل فإنها ستأتي في الفصول التالية ، حيث إن الغرض من هذا الفصل فقط هو تشويق القارئ للدخول في البرنامج ومعظم ذلك من خلال الصورة التفاعلية.

الفصل الثاني : قدمنا فيه الصورة الثانية من صور التعامل مع الماتلاب ، وهي صورة البرامج التي تسمى ملفات الإم أو الـ M files. الفصل الثالث : قدمنا فيه ماتلاب كلغة برمجة عامة مثل لغة الـ C^{++} حيث قدمنا فيه مكونات لغة ماتلاب والحلقات والشروط والكثير من تقنيات البرمجة العامة. الفصل الرابع : قدمنا فيه المصفوفات والمتجهات حيث إن برنامج ماتلاب يعد برنامج مصفوفات ومتجهات ، واستعرضنا في هذا الفصل الكثير من العمليات التي يمكن إجراؤها على المصفوفات في ماتلاب مثل عمليات الجمع والضرب والمحددات والعكس وغيرها الكثير. في الفصل الخامس : قدمنا أساسيات الرسم في ماتلاب حيث أن ماتلاب يحتوي على العديد من دوال الرسم ثنائي وثلاثي الأبعاد ، والتي تعتبر على درجة عالية جدا من الأهمية في استعراض نتائج أي برنامج.

في الفصل السادس : قدمنا تصميم شاشات التقابل أو المواجهة مع المستخدم والتي من خلالها يمكن عرض التطبيقات في صورة سهلة ومفيدة لكل المستخدمين الذين لا يعرفون الماتلاب. في الفصل السابع : قدمنا برنامج المحاكاة simulink بدرجة مبسطة للمستخدم المبتدئ في هذا المجال. الفصل الثامن : يشرح طرق إدخال وإخراج البيانات من وإلى الحاسب وكيفية عرض هذه البيانات. الفصل التاسع : يقدم الطرق الرمزية لإجراء بعض العمليات الرياضية ، مثل التفاضل والتكامل بطريقة رمزية وغير عددية والتى نحتاجها في الكثير من العمليات الرياضية.

الفصل العاشر: يقدم أساسيات معالجة الصور ويعد ماتلاب غنيا جدا بالدوال والخوارزميات التي تساعد في تحسين ومعالجة الصور، وفي الحقيقة فإنه يمكن إفراد كتاب بالكامل عن استخدام ماتلاب في معالجة الصور ولكننا قدمنا أساسيات الموضوع وتركنا الباقي للقارئ ليتابعه بنفسه. الفصل الحادي عشر: والأخير قدمنا فيه أساسيات معالجة الإشارات، وهذا الفصل يعد مقدمة لهذا الموضوع حيث أنه يمكن شرح هذا الموضوع في كتاب بالكامل، وتوجد في السوق العديد من الكتب الدراسية الموضوعة خصيصا لمعالجة الإشارات باستخدام ماتلاب؛ لذلك كان هذا الفصل مقدمه فقط عن الموضوع وتركنا الباقي للقارئ للمتابعة بنفسه.

كما ترى، فقد راعينا أن تكون فصول الكتاب عامة، وفي الموضوعات المستخدمة من قبل جميع التخصصات تقريبا وبخاصة الهندسية منها، فالكل يحتاج للرسم ويحتاج للتعامل مع الصور وإدخال وإخراج الإشارات من وإلى الحاسب كما يحتاج الجميع لطرق المحاكاة التي لاغنى عنها لأي تخصص.

ننصح القارئ الجديد على برنامج ماتلاب أن يبدأ بالخمسة فصول الأولى على الترتيب، فعليه أن يقرأها بتدبر وتمعن مع تنفيذ جميع البرامج الواردة فيها. بعد ذلك بالنسبة للستة فصول التالية فعليه أن يختار منها ما يريد أو يستمر في قراءتها هي الأخرى بالترتيب للحصول على أعظم فائدة من الكتاب.

في النهاية، نتمنى للقارئ الاستمتاع بما في هذا الكتاب ورحلة موفقة من أوله إلى آخره.

المؤلفون

المحتويات

الفصل الثاني: ملفات الإم

الفصل الثالث: أساسيات استخدام الماتلاب كلغة برمجة عامة

الفصل الرابع: المصفوفات في ماتلاب

الفصل الخامس: أساسيات الرسم في ماتلاب

الفصل السادس: شاشات التعامل مع المستخدم

الفصل السابع: برنامج المحاكاة سميولينك

الفصل الثامن: اكتساب البيانات

الفصل التاسع: الحسابات الرمزية في برنامج ماتلاب

الفصل العاشر: أساسيات استخدام الماتلاب في معالجة الصور الرقمية

الفصل الحادي عشر: معالجة الإشارات

مقدمة عن برنامج ماتلاب

(١,١) مقدمة

برنامج ماتلاب هو برنامج عالي المستوى والأداء للغة عالية المستوى والأداء أيضا بخاصة في الحسابات والتطبيقات التقنية، مثل الهندسة بجميع فروعها، والرياضيات، والفيزياء وغيرها. في هذا الفصل نعرض مكونات هذا البرامج كنظام برامجي ثم نشرح كيفية تشغيله. بعد ذلك نتطرق لشرح عام عن استخدام الماتلاب في صورته التفاعلية، والتي تمكننا من استخدامه كآلة حاسبة لإجراء بعض العمليات الحسابية البسيطة على المتغيرات أو بعض الدوال العامة. ثم نتناول بشيء من التفصيل كيفية طلب المساعدة في الماتلاب التي تعد من الخواص التي يتميز بها الماتلاب دون غيره من لغات البرمجة الأخرى. وأخيرا نتعرض سريعا لبعض الموضوعات – التي يسهل التعامل معها باستخدام الماتلاب دون غيره من لغات البرمجة– مثل رسم الدوال، التعامل مع الصور و الأصوات، والأرقام المركبة. هذه الموضوعات سيتم شرحها لاحقا فيما يلي من فصول هذا الكتاب.

(١,٢) ما هو الماتلاب ؟

برنامج ماتلاب عبارة عن وسط برمجة سهل الاستخدام يمكنك من استخدام هذه اللغة بصور متعددة. أبسط هذه الصور هي صورة الآلة الحاسبة التي يكون فيها التفاعل بين المستخدم والبرنامج من أسهل وأسرع ما يكون. أو كوسط برمجة يمكن للمستخدم فيه أن يكتب أي خوارزم أو برنامج يقوم بحل مشكلته ثم تنفيذ هذا الخوارزم واختباره بالكيفية التي يريدها مثله في ذلك مثل لغات البرمجة الشهيرة، مثل لغة C، والجافا وغيرها. وتتدرج الاستخدامات إلى إمكانية وضع البرنامج أو الخوارزم الذي صممته في صورة دالة من دوال ماتلاب يمكنك إضافتها إلى مكتبة ماتلاب وتنفيذها بمجرد كتابة اسمها كإحدى الدوال الداخلية في الماتلاب. علاوة على كل ذلك فتوجد في ماتلاب مكتبة من دوال الرسم والمحاكاة التي تمكنك من عرض نتائجك كصور ثنائية الأبعاد وثلاثية الأبعاد بسهولة تعجز عنها الكثير من البرمجيات في هذا المجال. إن استخدام ماتلاب كآلة حاسبة أو في صورته التفاعلية تعد من أهم مميزات ماتلاب والتي لا توجد في أي لغة برمجة أخرى حيث يمكن مباشرة حساب الجذر التربيعي مثلا لأي ثابت كما يمكن أيضا في هذه الصورة التفاعلية إجراء الكثير من العمليات الحسابية المعقدة مثل حل المعادلات التفاضلية وبالطبع ستحتاج لأكثر من سطر.

بجانب كل ذلك يحتوي ماتلاب على الكثير من المكتبات النوعية أو المتخصصة يسميها الماتلاب صناديق الأدوات tool boxes يحتوي كل منها على الكثير من الدوال المتخصصة في مجال معين، والتي تسهل على المستخدم التعامل معها من خلال هذه المكتبات المتخصصة. من هذه المكتبات والتي سيتم شرح بعضها (وذلك لصعوبة شرحها بالكامل في كتاب واحد) بالتفصيل في أماكن مخصصة في هذا الكتاب ما يلي :

- مكتبة المعلوماتية الطبية Bioinformatics tool box.
- مكتبة نظم الاتصالات Communication systems tool box.
- مكتبة نظم التحكم Control systems tool box.
- مكتبة اكتساب أو قراءة البيانات Data acquisition tool box .
- مكتبة قواعد البيانات Data base tool box.
- مكتبة تصميم المرشحات Filter design tool box .
- مكتبة المنطق الهلامي Fuzzy logic tool box .
- مكتبة الخوارزمات الجينية Genetic algorithms tool box.
- مكتبة معالجة الصور Image processing tool box .

وهناك الكثير من هذه المكتبات التي يذخر بها هذا البرنامج، والتي تعد من أهم مميزاته نظرا لما تقدمه من سهولة في البرمجة والتعامل للمتخصصين في كل هذه المجالات.

يحتوي برنامج ماتلاب على وسط برمجة آخر يستخدمه كل المهندسين وربما غيرهم أيضا وهو وسط برمجة المحاكاة Simulink. هذا الوسط يتعامل ليس من خلال أوامر وخوارزمات مكتوبة، ولكن من خلال صناديق أو بلوكات محاكاة. فيمكنك مثلا سحب صندوق مصدر إشارة تتحكم في تردد ومقدار هذه الإشارة من خلال خواصه، ثم تسحب صندوقا آخر يمثل مرشحاً تتحكم أيضا في خواصة، حيث يقوم هذا الصندوق بالسماح لبعض الترددات بالمرور للخرج ومنع بعضها الآخر، ثم في النهاية يمكنك أن تسحب صندوقا ثالثا يمثل سماعة تسمع من خلالها على سماعة الحاسب الذي تتعامل معه الصوت الخاص بالإشارة الخارجة من مصدر الإشارة، حيث يتم ذلك بالطبع بعد توصيل هذه البلوكات مع بعضها على التوالي. إن هذا مجرد مثال بسيط باستخدام ثلاثة بلوكات يعرفها الجميع، ولكن بالطبع يمكن تصميم أنظمة كاملة بهذه الطريقة ودراسة أدائها في هذا الوسط قبل البدء في تنفيذها. يحتوي الماتلاب

على العديد من المكتبات المتخصصة في كل المجالات في هذه الصورة من بلوكات المحاكاة. بالطبع، فإن وسط برمجة المحاكاة لن يتم شرحه بالكامل في هذا الكتاب، ولكن – إن شاء الله – سيفرد له كتاب خاص به في المستقبل القريب.

كلمة ماتلاب مأخوذة من التعبير "معمل المصفوفات **Matrix Laboratory**, MATLAB" وذلك لأن وحدة البيانات الأساسية فيه هي المصفوفة، فكل المتغيرات وكل الثوابت ينظر إليها الماتلاب على أنها مصفوفة، حتى أنك لو كتبت x=10 فإن ذلك ينظر إليه في جميع لغات البرمجة على أن هناك متغير x وضعت فيه القيمة أو الثابت ١٠، ولكن في الماتلاب يكون الوضع مختلفاً حيث ينظر الماتلاب للتعبير x=10 على أنه هناك متغير x وضعت فيه مصفوفة أحادية الأبعاد، أي مصفوفة من صف واحد وعمود واحد، أي مصفوفة بها عنصر واحد هو الثابت ١٠. هذه الطريقة الفريدة لتمثيل البيانات التي اتبعها الماتلاب سمحت بحل الكثير من المشاكل خاصة التقنية منها والتي تتعامل مع مصفوفات بيانات بأبعاد هائلة يصعب التعامل معها أو تأخذ وقتا كبيرا لحسابها ومن أمثلة ذلك الصور. فتخيل مثلا أن لديك صورتين أبعاد كل منهما ١٠٢٤×١٠٢٤ بكسل وتريد ضرب هاتين الصورتين في بعضهما، إن ذلك يتم في الماتلاب بسهولة جدا بالمقارنة باللغات الأخرى.

يتكون برنامج الماتلاب كنظام برامجي software system من خمسة أجزاء أساسية نذكرها فيما يلي:

١- **وسط البرمجة**: وهو مجموعة من الأدوات السهلة الاستخدام عن طريق النقر بالفأرة. وهو أول شاشة تظهر لك على سطح المكتب بمجرد النقر على أيقونة الماتلاب والدخول فيه، وهذه الشاشة مقسمة إلى أكثر من جزء، منها جزء خاص بمساحة التشغيل work space، ونافذة الأوامر command window، ومساحة تاريخ الأوامر command history وغيرها، والتي سيأتي شرحها بالتفصيل في الجزء التالي من هذا الفصل.

٢- مكتبة دوال ماتلاب : وهي مكتبة كبيرة تضم العديد من الدوال سابقة التجهيز التي يتم تنفيذها بمجرد كتابة اسمها ، وتغطي هذه المكتبة تقريبا جميع التخصصات العلمية بحيث إنه ما من دالة تريد إجراءها في معظم التخصصات العلمية إلا ستجدها جاهزة في مكتبة الماتلاب ، وما عليك لكي تنفذها إلا أن تكتبها بصورتها الصحيحة. تتغير هذه الدوال من الدوال البسيطة ، مثل المجموع ، والمتوسط ، وجيب التمام sine و cosine ، والتعامل مع الأرقام المركبة إلى الدوال المعقدة ، مثل ضرب المصفوفات ، وإيجاد معكوسها ، ودوال بيسيل ، ومحولات فورير وغير ذلك الكثير ، وسيتم الحديث عن هذه الدوال في أماكنها.

٣- لغة برمجة ماتلاب : وهي لغة عالية المستوى وحدتها هي المصفوفة تشمل كل العناصر المطلوبة لأي لغة برمجة ، مثل الشروط ، والحلقات ، وهياكل البيانات ، والإدخال ، والإخراج والبرمجة ، الموجهة بهدف مثلها في ذلك مثل لغة C ، والجافا حيث باستخدام هذه اللغة يمكن تصميم وبناء أعقد الخوارزمات. وهذه اللغة سيتم شرحها في فصل خاص بذلك.

٤- مكتبة الرسم : تضم هذه المكتبة العديد من الأوامر التي يمكن بها رسم أي دالة أو نتيجة في الأبعاد الثنائية أو الثلاثية بجانب أوامر قراءة ومعالجة وعرض الصور. كل هذا بجانب تصميم شاشات التفاعل مع المستخدم Graphical user interface, GUI وسيتم شرح ذلك في حينه أيضا.

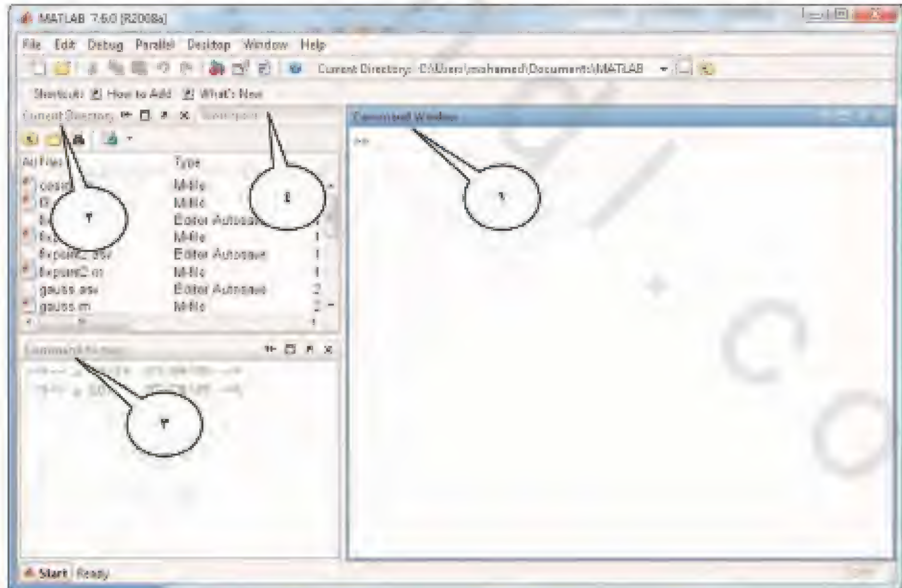
٥- برنامج تفاعل ماتلاب مع التطبيقات The matlab Application Program Interface, API : وهذه المكتبة تسمح للمستخدم بكتابة برامج بلغة C أو الفورتران وتوصيلها أو تنفيذها من خلال ماتلاب.

(١,٣) بدء التشغيل وسطح المكتب في ماتلاب

بالطبع سنفترض هنا أنه قد تم تثبيت برنامج ماتلاب على حاسبك، وسنفترض أنك قد قمت بتثبيت الإصدار ٧ وهو الذي نتعامل معه في هذا الكتاب، مع العلم أنه لا يوجد فرق كبير بين الإصدارات الأخيرة بدءا من الإصدار ٥.

من على سطح المكتب لبرنامج النوافذ انقر على أيقونة ماتلاب مرتين ستدخل فورا في شاشة وسط البرمجة الخاص به كما في شكل (١,١) وهذه الشاشة سنطلق عليها سطح مكتب الماتلاب.

للخروج من ماتلاب بعد الانتهاء من جلسة برمجة يمكنك النقر على الخيار Exit من قائمة الملفات File من على سطح المكتب. ويمكنك الخروج أيضا من ماتلاب بكتابة الأمر quit في نافذة الأوامر على سطح المكتب.



شكل (١,١). مكونات سطح مكتب الماتلاب.

تتكون شاشة سطح المكتب في ماتلاب من ٤ أجزاء أو نوافذ رئيسة كما في شكل (١.١) وهي كما يلي :

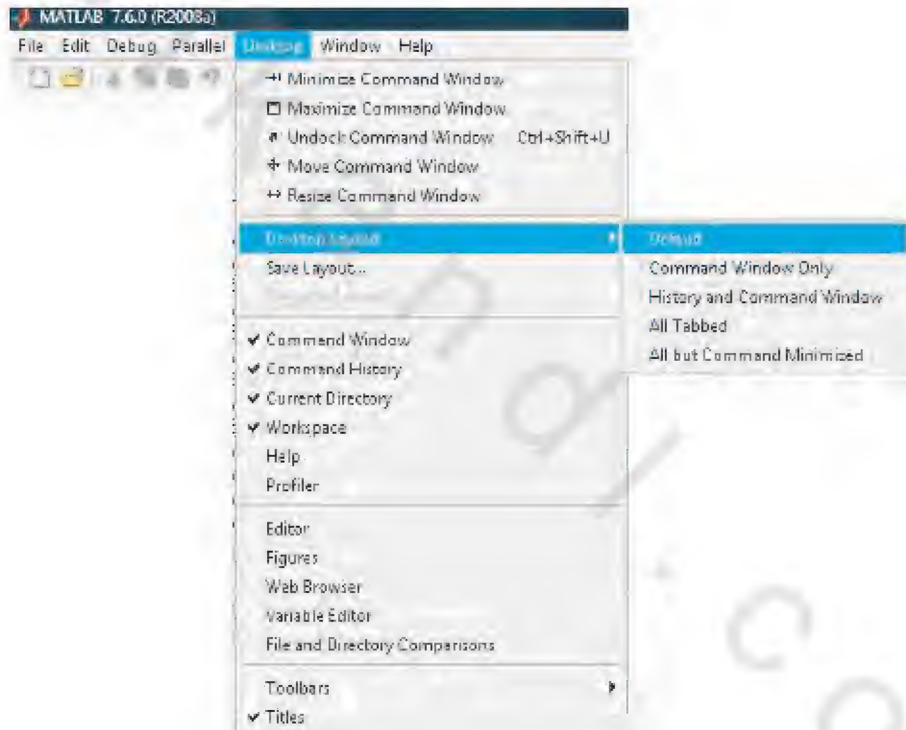
١- نافذة الأوامر Command window : وفيها نكتب الأوامر في حالة استعمال ماتلاب في الصورة التفاعلية أو كآلة حاسبة ، وتظهر فيها أيضا نتيجة تنفيذ أي أمر كما سنرى بعد قليل.

٢- نافذة المجلد الحالي Current directory : حيث تحتوي هذه النافذة محتويات المجلد الموجود فيه الماتلاب. ويمكن تغيير هذا المجلد باستخدام السهم النازل في شباك المجلد الحالي Current directory في شريط الأيقونات. بالنقر مرتين على أي ملف في هذا المجلد يتم فتحه.

٣- نافذة تاريخ الأوامر Command history : وتحتوي هذه النافذة على قائمة بكل الأوامر التي تم استخدامها في نافذة الأوامر من بداية جلسة تشغيل ماتلاب حتى نهاية الجلسة. هذه الأوامر تكون مرتبة وبالنقر على أي أمر فيها مرتين يظهر هذا الأمر في نافذة الأوامر ويتم تنفيذه. وسنتحدث عن ذلك مع الحديث بشيء من التفصيل عن نافذة الأوامر في الجزء التالي.

٤- نافذة مساحة العمل Work space : هذه النافذة تظهر بالتبادل مع نافذة المجلد الحالي حيث بالنقر على شريط مساحة العمل تظهر نافذة مساحة العمل وتختفي نافذة المجلد الحالي، وبالنقر على شريط المجلد الحالي مرة أخرى تظهر نافذة المجلد الحالي مرة أخرى وتختفي نافذة مساحة العمل. نافذة مساحة العمل تعرض جميع المتغيرات variables التي تم استخدامها في أي جلسة تنفيذ من جلسات الماتلاب. إننا نعني بكلمة جلسة هنا من بداية تشغيل ماتلاب حتى الخروج منه في أي مرة من المرات.

٥- نوافذ سطح المكتب للماتلاب الموجودة في شكل (١،١) هي التقسيمة التلقائية default لهذه النوافذ. يمكن تصغير مساحة كل نافذة أو تكبيرها أو إخفاء بعضها باستخدام الفأرة كما نفعل مع أي نافذة في برنامج النوافذ. كما يمكن التحكم في ترتيب هذه النوافذ وطريقة ظهورها بالكيفية التي يريدها المستخدم. بمجرد خروج المستخدم من أي جلسة وإغلاق ماتلاب فإن آخر وضع لهذه النوافذ يتم الاحتفاظ به.



شكل (١،٢). اختيار تقسيمة سطح مكتب الماتلاب.

يمكن التحكم في كيفية ظهور سطح المكتب في ماتلاب بالنقر على قائمة Desktop ومنها نختار Desktop Layout ثم نختار منها أي شكل نريد كما في شكل (١,٢). لاحظ أن الاختيار Default هو الاختيار التلقائي الذي يختاره لك ماتلاب وهو الموضح في شكل (١,١) وهو أفضل الاختيارات للكثير من المستخدمين. بالنقر على الاختيار Command Window Only ستظهر نافذة الأوامر فقط، والنقر على History and Command Window تظهر نافذة الأوامر ونافذة تاريخ الأوامر فقط، وبالنقر على All Tabbed ستظهر لك أحدى هذه النوافذ تملأ شاشة الحاسب مع أزرار tabs للنوافذ الأخرى بحيث يمكن التنقل بين هذه النوافذ بالنقر على الزر الخاص بكل منها، حاول تجربة ذلك. في أي لحظة يمكنك النقر على الاختيار التلقائي Default للعودة إلى الشكل التلقائي كما في شكل (١,١).

(١,٤) الماتلاب التفاعلي أو الماتلاب كآلة حاسبة

سنرى في هذا الجزء طريقة من طرق تشغيل ماتلاب، والتي يتميز بها عن جميع لغات البرمجة وهي تشغيله كآلة حاسبة أو بطريقة تفاعلية، حيث بمجرد كتابة الأمر والضرب على زر Enter يتم تنفيذ الأمر وإظهار النتيجة مباشرة. لكي نرى ذلك؛ ابدأ في تشغيل ماتلاب وانقر على نافذة الأوامر لكي تنفذ الأوامر التالية: (جميع الأوامر والعمليات التالية تم تنفيذها على ماتلاب ونسخها هنا، ولذلك فإنك لو نفذتها بنفس الطريقة ونفس التتابع فلابد أن تحصل على نفس النتائج).

١- العمليات الحسابية البسيطة

```
>> 2+3
ans =
     5
```

العلامة >> معناها أن ماتلاب ينتظر أمر بعد هذه العلامة، وقد قمنا بكتابة ٣+٢ أي أن الأمر هو جمع هذين الرقمين، وبمجرد أن نضغط الزر Enter سيرد عليك ماتلاب بالنتيجة ans=5 كما رأينا. فيما يلي سنكتب كل الأوامر بالخط الظاهر bold، والإجابة من الماتلاب ستكون بالخط العادي.

```
>> 3-2
ans =
    1
>> 3*2
ans =
    6
```

لاحظ في الأمر السابق أن * هي علامة الضرب.

```
>> 3/2
ans =
   1.5000
```

وعلامة القسمة هي الشرطة المائلة ناحية اليمين / ، هنا البسط على يسار العلامة والمقام على يمينها؛ ومن ثم فإن الأمر السابق سيقسم الرقم ٣ على الرقم ٢ والنتيجة كانت ١,٥.

```
>> 2\1
ans =
   0.5000
```

الشرطة المائلة ناحية اليسار تمثل عملية قسمة أيضا بحيث إن البسط هنا يكون على يمين العلامة والمقام على يسارها، وعلى ذلك فالأمر السابق يقسم ١ على ٢ لتكون النتيجة ٠,٥.

```
>> 2^3
ans =
    8
```

العلامة ^ هي علامة الأس، أي أن الأمر السابق يحسب ٢ أس ٣ لتكون النتيجة ٨.

• في أثناء كتابة أي أمر يمكنك استخدام جميع أزرار تصحيح النص، مثل أزرار السهم الأيمن right arrow، والسهم الأيسر والزر Del، والزر Backspace. لاحظ أن هذه الأزرار تساعد في التصحيح على خط أو سطر واحد فقط فليس هناك تحرك لأعلى أو لأسفل مثلا باستخدام السهم لأعلى أو السهم لأسفل.

• السهم لأعلى والسهم لأسفل يمكن استخدامها لإحضار أي أمر سبق كتابته بدلا من نعيد كتابته مرة أخرى. فمثلا إذا كنت قد كتبت الأوامر السابقة بنفس الترتيب السابق فإن بضرب زر السهم لأعلى سيحضر لك الأمر ٢^٣، وبضرب الزر Enter يتم تنفيذه. بضرب زر السهم العلوي مرة أخرى يظهر أمامك الأمر ١\٢، وإذا ضربت Enter يتم تنفيذه. وهكذا باستخدام السهم الأعلى والأسفل يمكن التجول في كل الأوامر السابقة حيث بعد ظهور أي أمر تريده يمكنك تنفيذه بضرب الزر Enter .

• ماتلاب به خاصية جميلة وهي الاستدعاء الذكي للأوامر السابقة، فبمجرد كتابة الأحرف الأولى من أي أمر ثم نضرب زر السهم العلوي يظهر الأمر بالكامل، حيث يمكنك تنفيذه بضرب الزر Enter. مثلا في الأوامر السابقة لو كتبنا >>٢+ ثم ضربنا السهم العلوي فإن الأمر تكتمل كتابته فيصبح >>٢+٣ وبضرب Enter تظهر النتيجة.

• بمناسبة القسمة، ماذا سيكون موقف ماتلاب عندما نقسم على الصفر؟. انظر للأمر التالي :

```
>> 1/0
ans =
  Inf
```

عند قسمة واحد على صفر كانت نتيجة القسمة هي Inf وهي اختصار لما لانهاية Infinity كما نعلم. هناك نتيجة أخرى قد تقابلها عند التعامل مع ماتلاب. انظر للمثال التالي :

```
>> 0/0
ans =
  NaN
```

هنا نقسم صفراً على صفر، والنتيجة بالطبع غير محددة Not a Number, NaN كما رأينا.

٢- المتغيرات

المتغير في أي لغة برمجة هو عنوان في الذاكرة يحمل اسم المتغير، ويمكن أن نضع أي قيمة في هذا المتغير باستخدام العلامة = . نبدأ الآن بإجراء هذه الأوامر على ماتلاب، حيث قمنا بنسخها هنا حتى يستطيع القارئ إعادتها والتدريب عليها:

```
>> a=2
a =
    2
```

بمجرد كتابة a=2 ثم نضرب Enter يرد الماتلاب بالإجابة، وهي أيضا a=2 مما يعنى أنه قد تم تخصيص القيمة ٢ للمتغير a، ويمكن الاستفادة بها كما في الأوامر التالية:

```
>> a+9
ans =
   11
```

هنا تم جمع الرقم ٩ مع المتغير a وكانت الإجابة ١١ ،

```
>> a*9
ans =
   18
```

وهنا تم ضرب a في ٩ وكانت الإجابة ١٨. لاحظ أن قيمة المتغير a لازالت تساوي ٢ ولم تتغير حتى الآن بالرغم من إجراء كل هذه العمليات السابقة على المتغير a . انظر لهذا الأمر:

```
>> a=a+10
a =
   12
```

الآن تغيرت قيمة a وأصبحت a=12. اكتب الأمر التالي:

```
>> b=4;
```

لاحظ أن الأمر b=4 هنا متبوعا بالفاصلة المنقوطة. وجود فاصلة منقوطة بعد أي أمر تمنع من عرض النتيجة فقط، بمعنى أنه يتم التنفيذ وتصبح قيمة المتغير b تساوي ٤ ولكن لم تعرض النتيجة كما كان في الأمر a=2.

```
>> c=a+b
c =
   16
```

آخر قيمة للمتغير a كانت ١٢ وبجمعها مع المتغير b تصبح النتيجة ١٦ التي يتم وضعها في المتغير c وعرضها كما رأينا. انظر للأمرين التاليين:

```
>> b
b =
    4
>> B
??? Undefined function or variable 'B'.
```

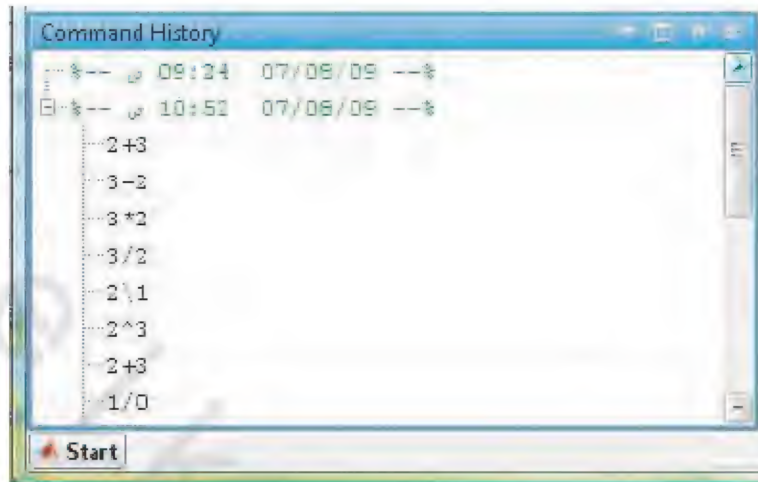
الأمر الأول b<< عرض قيمة المتغير b الموجودة عنده وهي ٤ ، بينما الأمر B<< لم يعرض أي نتيجة بل قال إن المتغير B متغير أو دالة غير محددة. إن هذا يقودنا إلى حقيقة مهمة وهي أن ماتلاب حساس لشكل الحرف. أي أن المتغير b يختلف عن المتغير B ، كل منهما يعد متغيراً مختلفاً ، وذلك على العكس من بعض لغات البرمجة التي لا تفرق بين الأحرف الصغيرة والكبيرة فصورة الحرف في هذه اللغات ليس لها قيمة.

استخدام الفاصلة المنقوطة تمكنك من كتابة أكثر من أمر في نفس السطر كما يلي:

```
>> x=4;  y=5;  z=x+y;
>> z
z =
    9
```

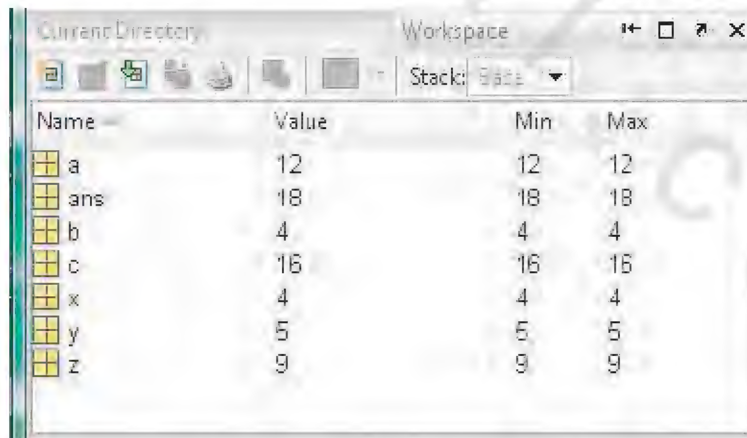
في السطر الأول تم وضع ٤ في المتغير x و ٥ في المتغير y ومجموع المتغيرين x و y في المتغير z ، ولم يتم عرض أي نتيجة بسبب وجود الفاصلة المنقوطة في نهاية كل أمر. بمجرد كتابة z تم عرض النتيجة z=9.

الآن يمكنك النظر إلى نافذة تاريخ تتابع الأوامر كما في شكل (١,٣). تلاحظ في هذه النافذة أن جلسة ماتلاب هذه كانت يوم ٢٠٠٩/٨/٧ الساعة العاشرة واثنتين وخمسين دقيقة صباحا ، ثم بعد ذلك تم عرض جميع هذه الأوامر بنفس طريقة كتابتها في نافذة الأوامر ، ولكن طبعا وكما نرى في الشكل دون عرض نتيجة تنفيذ هذه الأوامر. بالنقر على أي أمر في هذه النافذة يتم نقله إلى نافذة الأوامر وتنفيذه فحاول تجربة ذلك.



شكل (١,٣). نافذة تاريخ الأوامر.

شكل (١,٤) يبين نافذة مساحة العمل work space حيث تعرض هذه النافذة جميع المتغيرات التي تم استخدامها في جلسة البرمجة الحالية مرتبة بحسب استخدامها في نافذة الأوامر. هذه النافذة تعرض أيضا آخر قيمة أخذها كل متغير وأقل قيمة وأكبر قيمة أيضا لهذا المتغير.



Name	Value	Min	Max
a	12	12	12
ans	18	18	18
b	4	4	4
c	16	16	16
x	4	4	4
y	5	5	5
z	9	9	9

شكل (١,٤). نافذة مساحة العمل work space.

نلاحظ من هذا الشكل أيضا أن نتيجة أي عملية حسابية التي تكتب على الصورة >>ans=5 مثلا ؛ أن ans هو متغير أيضا من إنشاء ماتلاب ، ويمكن التعامل معه كأي متغير كما يلي :

```
>> x=4; y=6;
>> x*y
ans =
   24
>> ans=ans/2
ans =
   12
```

٣- مع الدوال الحسابية

برنامج ماتلاب لديه مكتبة ذاخرة بالدوال الحسابية التي قد تخطر على بالك ، ويمكن النداء عليها وتنفيذها ببساطة كما سنرى :

بالنسبة للدوال المثلثية مثل sin و cos فإن ماتلاب يتوقع وضع الزوايا بالرادين وليس بالدرجات فمثلا الزاوية ٣٠ درجة تكتب على الصورة 30*pi/180=pi/6 حيث pi=π.

```
>> sin(pi/6)
ans =
   0.5000
>> cos(pi/6)
ans =
   0.8660
```

اللوغاريتمات للأساس e :

```
>> log(10)
ans =
   2.3026
```

بينما اللوغاريتم للأساس ١٠ فيكتب على الصورة :

```
>> log10(10)
ans =
    1
```

يحتوي ماتلاب على العديد من الدوال الحسابية التي نقدم بعضها وأكثرها استخداما في جدول (١.١) فحاول تجربة هذه الدوال أو على الأقل ما يخصك منها.

جدول (١,١). قائمة ببعض الدوال الحسابية الشهيرة.

مسلسل	شرح الدالة		الدالة كما في ماتلاب
١	جيب التمام العكسي، النتيجة بالتقدير الدائري	Inverse cosine; result in radians	acos
٢	جيب التمام العكسي، النتيجة بالدرجات	Inverse cosine; result in degrees	acosd
٣	الجيب العكسي، النتيجة بالتقدير الدائري	Inverse sine; result in radians	asin
٤	الجيب العكسي، النتيجة بالدرجات	Inverse sine; result in degrees	asind
٥	الظل العكسي، النتيجة بالتقدير الدائري	Inverse tangent; result in radians	atan
٦	الظل العكسي، النتيجة بالدرجات	Inverse tangent; result in degrees	atand
٧	جيب التمام، النتيجة بالتقدير الدائري	Cosine of argument in radians	cos
٨	جيب التمام، النتيجة بالدرجات	Cosine of argument in degrees	cosd
٩	الجيب، النتيجة بالتقدير الدائرى	Sine of argument in radians	sin
١٠	الجيب، النتيجة بالدرجات	Sine of argument in degrees	sind
١١	الظل، النتيجة بالتقدير الدائري	Tangent of argument in radians	tan
١٢	الظل، النتيجة بالدرجات	Tangent of argument in degrees	tand
١٣	الأس الطبيعي e	Exponential	exp
١٤	اللوغاريتم الطبيعي للأساس e	Natural logarithm	log
١٥	اللوغاريتم للأساس 10	Common (base 10) logarithm	log10
١٦	الجذر التربيعي	Square root	sqrt
١٧	القيمة المطلقة	Absolute value	abs
١٨	التقريب لأعلى رقم صحيح	Round toward infinity	ceil
١٩	التقريب لأقل رقم صحيح	Round toward zero	fix
٢٠	التقريب لأقل رقم صحيح	Round toward minus infinity	floor
٢١	الباقي من القسمة	Remainder after division	rem
٢٢	التقريب لأقل رقم صحيح	Round to nearest integer	round

يمكن الحصول على قائمة كاملة بهذه الدوال من قائمة المساعدة Help ثم الدخول على MATLAB ثم Function Reference ثم Mathematics ثم Elementary Math حيث سترى عرضا كاملا للكثير من الدوال المستخدمة في ماتلاب مبوبة وظيفيا وأبجديا.

٤- طلب المساعدة في ماتلاب

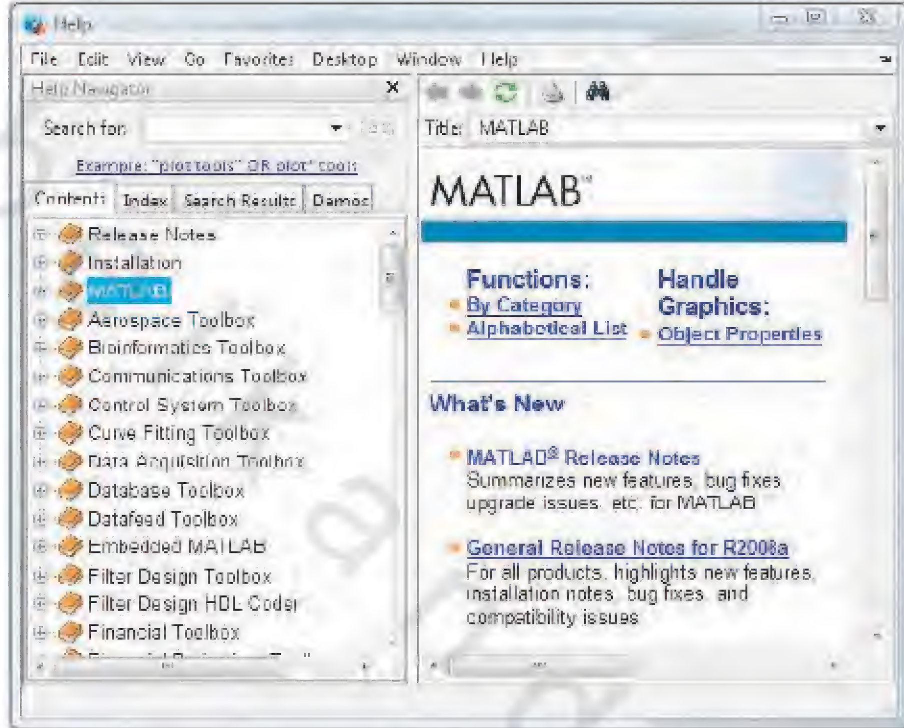
ماتلاب لديه نظام مساعدة نشك أنه موجود في أي لغة من لغات البرمجة الأخرى، حيث يمكنك الاعتماد على هذا النظام للمساعدة في تعلم ماتلاب بنفسك. يمكن طلب هذه المساعدة بأكثر من طريقة:

• في شريط القوائم هناك قائمة المساعدة Help كما في شكل (١,٥) ومنها يتم النقر على Product Help F1 حيث تفتح أمامك شكل (١,٦) والذي يمثل فهرساً كاملاً لجميع موضوعات ماتلاب بلا استثناء، والتي يمكنك الحصول على مساعدة لها.

• كذلك يمكن الحصول على شاشة المساعدة الموضحة في شكل (١,٦) عن طريق النقر على أيقونة المساعدة (؟) الموجودة في شريط الأدوات.



شكل (١,٥). قائمة المساعدة Help.

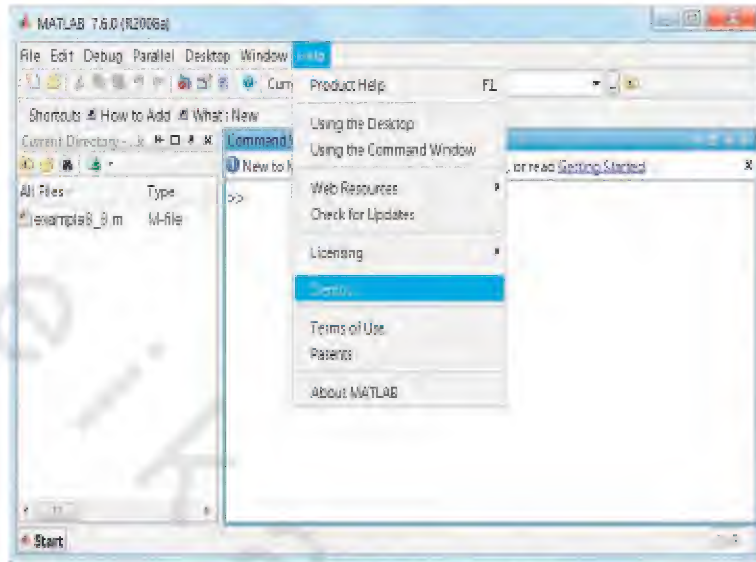


شكل (١,٦). شاشة المساعدة.

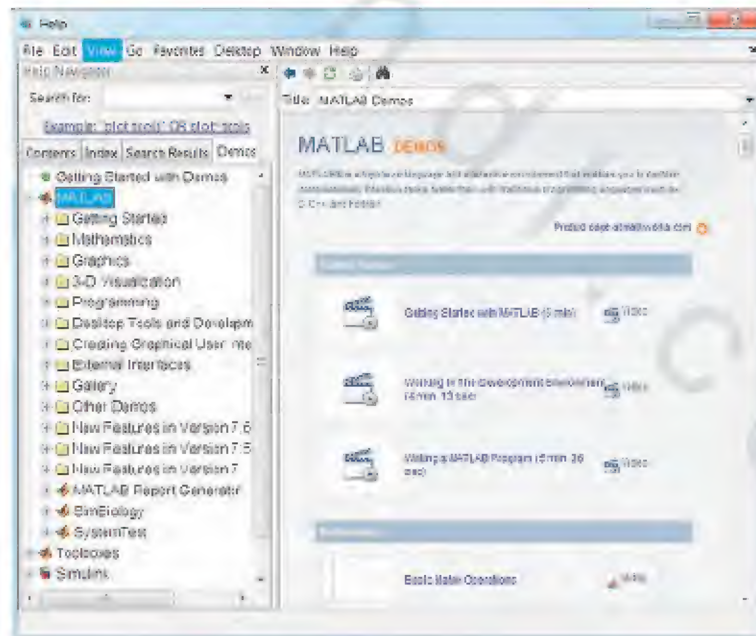
• يوفر الماتلاب بعض الفيديوهات المهمة التي تساعد المستخدم على البدء في التعامل مع الماتلاب، والتي يمكن الحصول عليها عند النقر على Demos من قائمة المساعدة Help كما في شكل (١,٧) وشكل (١,٨).

• كذلك يمكن الحصول على شكل (١,٨) مباشرة عند كتابة الأمر:

```
>> demo matlab
```



شكل (١,٧) قائمة المساعدة Help.



شكل (١,٨) نافذة توضح الفيديوهات التي يوفرها الماتلاب.

• هناك أيضا إمكانية الحصول على معلومات عن طريق كتابة أي عنوان تريده في خانة البحث (search) الموجودة في شكل (١.٦) وفي الحال سيعرض ماتلاب كل ما يتعلق بهذا العنوان.

• يمكن أيضا الحصول على مساعدة عن أي دالة مباشرة عن طريق كتابة أمر help في نافذة الأوامر كما في المثال التالي لطلب المساعدة عن الدالة sind :

```
>> help sind
 SIND   Sine of argument in degrees.
   SIND(X) is the sine of the elements of X, expressed in degrees.
   For integers n, sind(n*180) is exactly zero, whereas sin(n*pi)
   reflects the accuracy of the floating point value of pi.
   Class support for input X:
     float: double, single
   See also asind, sin.
   Overloaded methods:
     distributed/sind
   Reference page in Help browser
     doc sind
```

الأمر help يستخدم مع اسم الدالة لطلب المساعدة عن دالة معينة مثلا مدخلاتها أو مخرجاتها أو حتى كيفية استخدامها والمتغيرات فيها. أما في حالة عدم تذكر اسم الدالة أو عند البحث عن دالة تقوم بعمل محدد فإننا نستخدم الأمر lookfor لاحظ أن الاسم يكتب ككلمة واحدة. والذي يبحث عن مجموعة أحرف في السطر الأول لكل ملفات الإم الموجودة في الماتلاب ويعطيك اسم الدوال المحتوية على هذه الكلمة. المثال التالي يوضح نتيجة البحث عن دالة لحساب المتوسط.

```
>> lookfor average
Localavfit       - Construct "average fit" model
mean             - Average or mean value.
gcrma            - performs GC Robust Multi-array Average (GCRMA) background
gcrmabackadj     - performs GC Robust Multi-array Average (GCRMA) background
movavg           - Leading and lagging moving averages chart.
.
.
.
mean2            - Average or mean of matrix elements.
```

نلاحظ أن الماتلاب أعطى جميع الدوال التي تقوم بعمل المتوسط وعلى المستخدم تحديد الدالة التي يريد استخدامها.

٥- طريقة عرض الثوابت والمتغيرات

كل مستخدم يفضل قراءة القيمة العددية للثوابت والمتغيرات بطريقة معينة. فالبعض يفضل عرض الثوابت حتى أقصى دقة في ماتلاب وهي ١٦ خانة عشرية والبعض لا يريد العرض بهذه الدقة الزائدة التي لا يكون هناك حاجة لها في الكثير من التطبيقات. انظر للأمر format التالي :

```
>> format long
>> exp(1)
ans =
  2.718281828459046
```

الأمر format long يعرض النتيجة في ١٦ خانة عشرية كما رأينا. لاحظ أنه بعد كتابة الأمر format long ستظل طريقة العرض هذه باقية إلى أن يتم تغييرها بأمر آخر كما يلي :

```
>> format short
>> exp(1)
ans =
   2.7183
```

حيث تم العرض في صورة خمس خانات عشرية بدلا من ١٦. يمكن عرض النتائج في الصورة الأسية التالية :

```
>> format short e
>> exp(1)*10
ans =
 2.7183e+001
```

هنا تم العرض في صورة خمس خانات عشرية مضروبة في أس حيث $e+001=10^1$. هناك الكثير من طرق العرض الأخرى يمكنك مراجعتها عن طريق طلب المساعدة help format. تذكر أن دقة ماتلاب هي ١٦ خانة عشرية وعند العرض في صورة خمس خانات عشرية، فإن دقة الرقم المخزنة في الذاكرة (١٦خانة) لا تتغير، الذي يتغير فقط هو طريقة عرض الرقم أو الثابت على الشاشة.

٦- بعض الدوال العامة المفيدة في ماتلاب

هناك بعض الدوال العامة التي تساعد في تشغيل ماتلاب، ولكنها لا تنفذ أي عملية حسابية منها ما يلي :

```
>> whos
 Name    Size        Bytes   Class    Attributes
 a        1x1          8      double
 ans      1x1          8      double
 b        1x1          8      double
 c        1x1          8      double
 x        1x1          8      double
 y        1x1          8      double
 z        1x1          8      double
```

حيث تعرض هذه الدالة جميع المتغيرات التي تم استخدامها في نافذة مساحة العمل work space في صورة جدول - كما رأينا - يبين اسم المتغير وحجمه، حيث - كما نعلم - إن كل متغير يكون عبارة عن مصفوفة، وكل المتغيرات التي تعاملنا معها حتى الآن عبارة عن مصفوفة أحادية أي لها صف واحد وعمود واحد. كذلك يعرض الجدول عدد البايتات التي يشغلها كل متغير ونوعه، وسيأتي الحديث عن أنواع المتغيرات لاحقا. يمكن أيضا إجراء هذه الدالة على متغير واحد كما يلي :

```
>> whos x
 Name    Size       Bytes  Class    Attributes
   x      1x1          8  double
```

الدالة clear تمسح متغيراً أو متغيرات من مساحة العمل (work space) كما يلي :

```
>> clear x
>> x
??? Undefined function or variable 'x'.
```

هنا تم مسح المتغير x من الذاكرة بالأمر clear x وبعدها تم السؤال عن نفس المتغير فكانت إجابة ماتلاب بأن هذا المتغير غير موجود. الدالة clear بدون كتابة أي متغير تمسح جميع المتغيرات الموجودة في مساحة العمل من الذاكرة.

بعد جلسة ماتلاب طويلة تكون نافذة الأوامر مليئة بالأوامر والإجابات، وفي بعض الأحيان نريد التخلص من كل هذه الكتابات لتنظيف نافذة الأوامر. الدالة clc تقوم بذلك حيث تمسح محتويات نافذة الأوامر وتضع دليل الكتابة cursor في أول الصفحة. تذكر أن الدالة clc لا تمسح المتغيرات من الذاكرة، إنها فقط تنظف نافذة الأوامر كما يلي:

```
>> clc
>> a
a =
   12
```

حيث هنا تم تنفيذ الأمر clc فتنظفت نافذة الأوامر، وبعدها سألنا عن المتغير a فعرض لنا ماتلاب قيمة هذا المتغير مما يؤكد أن المتغير a لم يمسح من الذاكرة.

```
>> date
ans =
08-Aug-2009
```

هذه الدالة تعرض التاريخ في صورة اليوم والشهر والسنة كما رأينا.

```
>> calendar
          Aug 2009
   S    M   Tu    W   Th    F    S
   0    0    0    0    0    0    1
   2    3    4    5    6    7    8
   9   10   11   12   13   14   15
  16   17   18   19   20   21   22
  23   24   25   26   27   28   29
  30   31    0    0    0    0    0
```

الدالة calendar تعرض نتيجة الشهر الحالي كما رأينا.

الدالة clock تعرض لك التاريخ والوقت في متجه بالترتيب التالي من اليسار لليمين: السنة ٢٠٠٩ مثلا، ثم الشهر (شهر ٨ مثلا)، ثم اليوم، ثم الساعة، والدقيقة والثانية كما يلي:

```
>> fix(clock)
ans =
     2009        8       16        8       46        5
```

ينصح باستخدام fix قبل الأمر clock حتى يتم عرض الأرقام صحيحة بدون كسور. هنا كانت الإجابة عام ٢٠٠٩ شهر ٨ يوم ١٦ الساعة ٨ والدقيقة ٤٦ وخمس ثوان.

٧- دوال الرسم في ماتلاب

من أهم المميزات الخاصة بماتلاب أنه غني جدا بدوال الرسم التي تمكنك من رسم أي دالة في بعدين أو ثلاثة أبعاد. ولأهمية دوال الرسم فلقد تم إفراد الفصل الخامس بالكامل لشرح أساسيات الرسم في ماتلاب.

٨- التعامل مع الصور

التعامل مع الصور ومعالجتها وتحسينها من أفضل محتويات ماتلاب ومن أكثر التطبيقات التي يتم التعامل معها من خلال مكتبة خاصة بمعالجة الصور تضم الكثير من الأوامر الخاصة بذلك والتي لا مجال لشرحها هنا لأن مجال معالجة الصور تفرد له مراجع وكتب خاصة به. الأوامر التالية تعرض الصور الموجودة في شكل (١,٩). الأمر الأول imread() يقرأ الصورة المكتوب مسارها بين علامتي التنصيص ويضع هذه الصورة في متغير اسمه I. إذا حاولت تنفيذ هذا البرنامج على أي صورة على جهازك فتذكر أن تضع المسار الصحيح للصورة التي تريد التعامل معها. الأمر الثاني rgb2gray() يحول الصورة من ألوان إلى رمادي ويسميها J. الأمر الثالث يغير أبعاد الصورة ليجعلها ٢٥٦×٢٥٦ بكسل. الأمر الرابع يعرض الصورة، والأمر الخامس يضيف ضوضاء لهذه الصورة، والأمر السادس يعرض الصورة بعد إضافة الضوضاء لها، والأمر التالي medfilt2() يرشح الصورة للتخلص من هذه الضوضاء، والأمر الأخير يعرض الصورة بعد تخليصها من الضوضاء. انظر لشكل (١,٩) لترى هذه الصور وتلاحظ الفرق بينها.

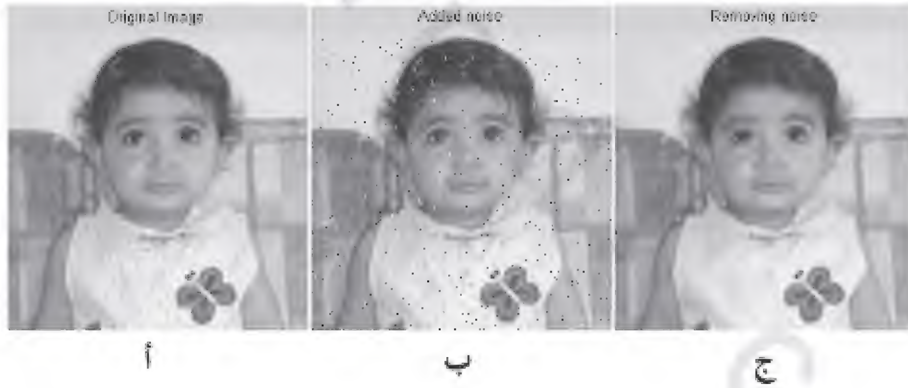
```
>> I=imread('d:\aseel\aseel256.jpg');
>> J = rgb2gray(I);
>> J = imresize(J,[256 256]);
>> imshow(J);title('Original Image')
>> J = imnoise(J,'salt & pepper',0.02);
>> figure, imshow(J);title('Added noise')
>> L = medfilt2(J,[3 3]);
>> figure, imshow(L);title('Removing noise')
```

٩- الأصوات في ماتلاب

لقد رأينا بعض ما لدى ماتلاب من التعامل مع الصور ، فماذا لديه عن التعامل مع الأصوات؟ نفذ الأمرين التاليين واستمع :

```
>> load handel
>> sound(y,Fs)
```

حيث إن الأمر الأول ينادى على ملف صوتي ، والثاني يرسل إشارة هذا الملف إلى سماعة جهاز الحاسب. هناك الكثير من ملفات الصوت الموجودة مع الماتلاب تلقائيا يمكنك سماعها ، ومنها الملفات train ، و splat ، و chirp ، و gong ، و laughter فحاول تحميلها بالأمر load ثم سماعها بالأمر sound كما في الأمرين السابقين. يمكنك تسجيل أي ملف صوتي من عندك والنداء عليه بالمسار الصحيح وسماعه.



شكل (١.٩) تأثير مرشح الوسط median filter على ضوضاء الملح والفلفل أ- الصورة الأصلية ب- الصورة مضافا إليها الضوضاء ج- الصورة بعد إزالة الضوضاء.

١٠- الأرقام المركبة في ماتلاب

آخر ما سنقدمه هنا من عرض لقدرات الماتلاب هو عرض سريع للأرقام المركبة في ماتلاب. كما نعلم، فإن أي رقم مركب يتكون من جزأين ؛ جزء حقيقي وآخر تخيلي. الجزء التخيلي يسبقه ماتلاب بالحرف i أو j (تذكر أن j=√-1)، انظر للعمليات التالية :

```
>> a=5+j*6
a =
  5.0000 + 6.0000i
>> b=3+i*4
b =
  3.0000 + 4.0000i
>> c=a+b
c =
  8.0000 +10.0000i
>> c=a*b
c =
 -9.0000 +38.0000i
>> exp(a)
ans =
 1.4250e+002 -4.1469e+001i
>> z=real(b)
z =
    3
```

الأمر real() يحسب القيمة الحقيقية لأي متغير مركب،

```
>> y=imag(ans)
y =
 -41.4689
```

كما أن الدالة imag() تعرض القيمة التخيلية لأي متغير، كما هو واضح من النتائج.

كما رأينا فإن ماتلاب يتعامل مع الأرقام والمتغيرات المركبة بسهولة مثل الأرقام الحقيقية تماما. في أي لغة من لغات البرمجة الأخرى لابد من عمل برنامج لإجراء أي عملية حسابية على المتغيرات المركبة.

الفصل الثاني

ملفات الإم

(٢,١) مقدمة

إن التعامل مع ماتلاب بالطريقة التفاعلية التي رأيناها في الفصل الأول سهلة ومباشرة وبالذات في حالة التطبيقات القصيرة التي تتطلب سطرا واحدا أو حتى عددا قليلا من الأسطر أو من الأوامر التي يتم كتابتها في نافذة الأوامر command window. عندما تكون في حاجة إلى برنامج يتكون من العديد من الخطوات التي تريد تنفيذها ليس لمرة واحدة، ولكن للعديد من المرات وفي كل مرة بمعاملات أو بثوابت مختلفة، فإن تنفيذ مثل هذا البرنامج من نافذة الأوامر command window تكون صعبة وغير عملية. في هذا الفصل نشرح كيفية التعامل مع الماتلاب بطريقة أخرى "غير الطريقة التفاعلية"، حيث سنوضح كيفية كتابة وتنفيذ برنامج (مجموعة من الأوامر في ملف). ثم بعد ذلك نتطرق لكيفية تغيير المعاملات في البرنامج من ثابتة إلى متغيرة. وينتهي الفصل بشرح كيفية كتابة دالة لتقوم بوظيفة محددة.

لكي نفهم ذلك تعال نفترض، كمثال، أننا نريد استخدام ماتلاب في حساب جذري معادلة من الدرجة الثانية على الصورة:

$ax^2+bx+c=0$

جذرا هذه المعادلة – كما نعلم من أساسيات الرياضيات – يعطيان بالمعادلات التالية :

x_1 = -b/(2*a) + sqrt(b^2 - 4*a*c)/(2*a)

x_2 = -b/(2*a) - sqrt(b^2 - 4*a*c)/(2*a)

الآن سنستخدم نافذة مساحة البرمجة لحساب هذين الجذرين بفرض القيم التالية للثوابت : a=1 و b=5 و c=6 كما يلي :

```
>>  a = 1
a =
    1
>>  b = 5
b =
    5
>> c = 6
c =
     6
>> x1 = -b/(2*a) + sqrt(b^2 - 4*a*c)/(2*a)
x1 =
     -2
>> x2 = -b/(2*a) - sqrt(b^2 - 4*a*c)/(2*a)
x2 =
     -3
```

الآن افترض أننا نريد حساب الجذرين x_1 و x_2 ولكن لثوابت مختلفة ، ولتكن a=100 و b=50 و c=70. في هذه الحالة وفي نافذة الأوامر سنكتب قيم هذه الثوابت ، ثم نكتب (أو باستخدام أزرار الأسهم يمكن استدعاء) معادلة الجذر x_1 وتنفيذها للحصول على قيمة x_1 ثم استدعاء معادلة x_2 وحساب قيمته.

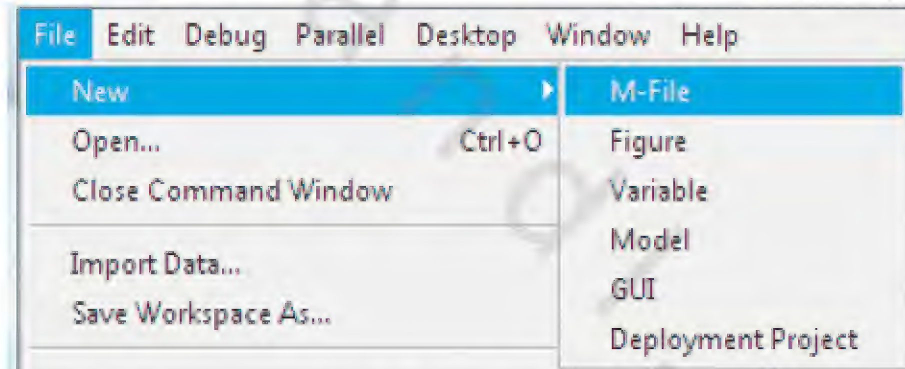
كما نرى، فإن هذه العملية ستحتاج إلى وقت كبير وبالذات في حالة البرامج الطويلة، كما أنها مناسبة طالما أننا نتعامل في نفس جلسة ماتلاب، فإذا تم إغلاق ماتلاب والدخول مرة ثانية ربما تحتاج لإعادة كتابة كل هذه الأوامر مرة ثانية؛ لأن عملية استدعاء هذه الأوامر مرة أخرى من نافذة الأوامر أو نافذة تاريخ الأوامر command history ستكون صعبة بالذات مع الاستعمال المتكرر لماتلاب، كما أن تخزين نسخة من خطوات الحل كبرنامج أو خواريزم يتم النداء عليه وتنفيذه كما في لغات البرمجة الأخرى ليست ممكنه بهذا الوضع؛ لذلك كان الحل لهذه المشكلة هو استخدام ملفات إم M files التي سنتعرف عليها في الجزء التالي.

(٢,٢) ملفات الإم

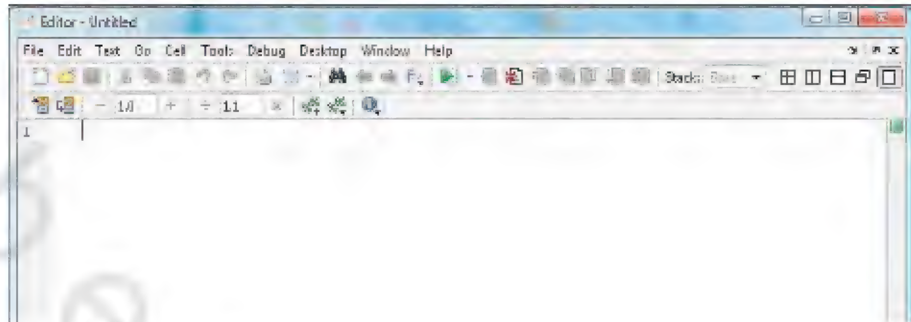
ملفات الإم في الماتلاب هي ملفات نصية نكتب فيها الأوامر كنصوص وبالترتيب الذي نريده ثم نخزن هذا الملف بأي اسم نريد، ولكن امتداد هذا الملف لابد أن يكون .m . بعد ذلك ومن نافذة الأوامر يمكن تنفيذ هذا البرنامج بمجرد كتابة اسم هذا الملف وضرب الزر Enter، وهذا بالطبع إذا كان البرنامج خالياً من الأخطاء اللغوية، حيث إن ماتلاب يقوم في البداية بمراجعة البرنامج قبل تنفيذه واستخراج ما به من أخطاء ويطالبك بتصحيح هذه الأخطاء قبل تنفيذ البرنامج. وبذلك يصبح البرنامج مخزنا في الذاكرة ويمكن النداء عليه وتنفيذه في أي وقت، ويمكن أن تكون لديك مكتبة بالبرامج الخاصة بك التي تحل الكثير من المسائل أو المشاكل التي تتعامل معها في مجال تخصصك أو بحثك.

لكي نجعل الأمور أكثر وضوحا سنكتب خطوات حساب جذري المعادلة من الدرجة الثانية في صورة ملف إم وننفذها كما يلي :

١- افتح ملف إم جديد. يمكن فتح ملف الإم من أي محرر نصوص مثل الـ word وكتابة النص فيه وتخزنه بالامتداد .m كما ذكرنا، ولكن ماتلاب لديه محرر نصوص خاص به يمكن أن نفتح منه ملف إم كالتالي. من قائمة ملفات File تحرك بالماوس على القائمة الفرعية New، ثم اختر منها M-File وانقر عليها كما في شكل (٢.١)، وفي الحال ستفتح أمامك ملفاً نصياً script، حيث ستجد أمامك كما في شكل (٢.٢) مساحة لتحرير أو كتابة النص الجديد الذي تريده مع قائمة بكل ما تحتاج إليه من وسائل تحرير النصوص.

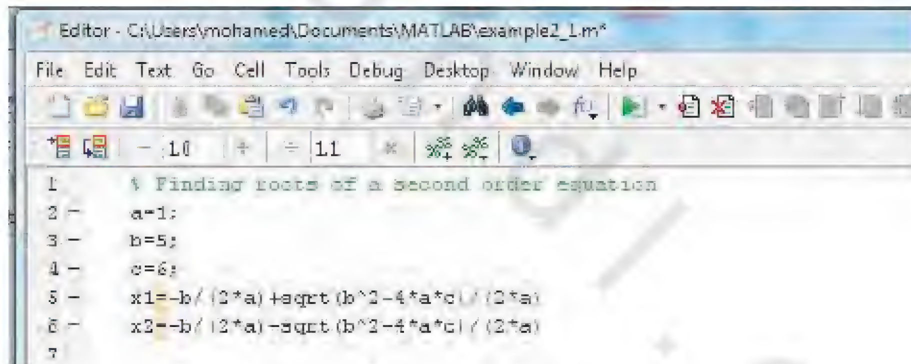


شكل (٢.١). فتح ملف إم M file.



شكل (٢,٢). ملف إم جديد.

٢- في مساحة التحرير ابدأ بكتابة البرنامج التالي، وهو نفس خطوات حساب جذري المعادلة من الدرجة الثانية مع بعض الإضافات البسيطة كما في شكل (٢,٣).



شكل (٢,٣). برنامج حساب الجذور في ملف إم باسم example2_1.

٣- في شكل (٢,٣) السطر الأول من البرنامج يبدأ بالحرف ٪ وهذا الحرف في ماتلاب يعنى أن باقي هذا السطر يعد تعليقا وليس أمراً قابلاً للتنفيذ ولذلك سنستخدم هذا الحرف بكثرة فيما بعد في كتابة أي تعليق في أي مكان من البرنامج.

٤- بعد الانتهاء من كتابة جميع خطوات البرنامج كما في شكل (٢،٣) نقوم بتخزينه بأي اسم نريد، وقد اخترنا هنا الاسم example2_1.m، وكما ذكرنا من قبل فأنت حر في اختيار اسم الملف أو البرنامج، فقط لابد أن يكون امتداده هو .m .

٥- الآن من نافذة الأوامر command window اكتب example2_1 ثم اضرب Enter ستظهر لك الإجابة، وهي جذور المعادلة كالتالي:

```
>> example2_1
x1 =
   -2
x2 =
   -3
```

وبذلك أصبح البرنامج example2_1 موجودا في الذاكرة يمكنك النداء عليه في أي وقت من نافذة الأوامر لتنفيذه بمجرد كتابة اسم البرنامج. كذلك يمكن تنفيذ ملف امتداده .m من الأيقونة الموجودة على المحرر (editor).

٦- من قائمة الملفات File يمكنك فتح الملف example2_1 وتعديله أو الإضافة إليه كيفما شئت ووقتما شئت. وبالمناسبة، سنجرى التعديل التالي على البرنامج لنجعله أكثر تفاعلية مع المستخدم ولكي يحسب جذري المعادلة لأي قيمة من الثوابت a و b و c:

```
% Finding roots of a second order equation
a=input('write the value of a: ');
b=input('write the value of b: ');
c=input('write the value of c: ');
x1=-b/(2*a)+sqrt(b^2-4*a*c)/(2*a)
x2=-b/(2*a)-sqrt(b^2-4*a*c)/(2*a)
```

الجديد هنا هو الأمر input() والذي عند تنفيذه يكتب العبارة الموجودة بين علامتي التنصيص 'write the value of a : ' في نافذة الأوامر ويقف في انتظارك أن تكتب قيمة المتغير a، حيث بمجرد كتابة قيمة a وضرب الزر Enter يذهب للأمر التالي، وهكذا حتى يكتمل البرنامج. الآن في نافذة الأوامر سنكتب example2_1 وننفذ البرنامج في صورته الجديدة كما يلي:

```
>> example2_1
write the value of a: 1
write the value of b: 5
write the value of c: 6

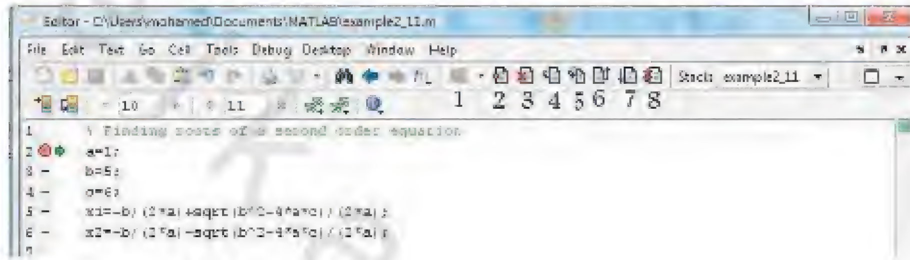

x1 =
    -2
x2 =
   -3
```

عندما تقوم بتخزين ملف الإم بأي اسم عليك اختيار اسم غير مستخدم في الدوال سابقة البناء في ماتلاب، وعلى العموم إذا حاولت تخزين ملف إم جديد بنفس الاسم لأي ملف سابق البناء في ماتلاب، فإن ماتلاب لن يقبله منك.

شركة MathWork المنتجة لبرنامج ماتلاب توفر موقعا على الإنترنت يمكنك الدخول عليه للاطلاع على ملفات الإم المختلفة من إعداد وتصميم مبرمجي ماتلاب من شتى أنحاء العالم الذين يعرضون هذه الملفات للاستخدام العام ويتبادلونها بينهم، ولتكن واحدا منهم. الموقع هو:

http://www.mathworks.com/matlabcentral/fileexchange/index.jsp

عندما يكون البرنامج طويلا جدا، وعند تنفيذه قد يحدث خطأ فيه، فإن عملية تحديد مكان الخطأ تكون صعبة؛ ولذلك فقد قدم ماتلاب أكثر من وسيلة تساعد المبرمج على تحديد هذه الأخطاء وهذه العملية تسمى في عالم البرمجة debugging أو استخراج الأخطاء. شكل (٢،٤) يبين قائمة الأدوات في محرر نصوص ملفات الإم. سنذكر هنا خطوات هذه العملية:



شكل (٢،٤) عملية تحديد الأخطاء debugging في ملفات الإم.

١- قبل تنفيذ البرنامج لابد من تخزين أي تغيرات تم عملها فيه، وبدون تخزين للبرنامج لا يمكن تنفيذه.

٢- يمكن تنفيذ البرنامج بالكامل إما من مساحة العمل work space بكتابة اسم البرنامج ثم ضرب الزر Enter. وإما من نافذة كتابة نص الملف إم كما في شكل (٢،٤) بالضرب على أيقونة زر التنفيذ رقم ١. بهاتين الطريقتين يتم تنفيذ البرنامج من أوله لآخره.

٣- في معظم الحالات نشك بوجود خطأ عند أمر معين ونريد تنفيذ البرنامج حتى هذا الأمر ونتوقف عنده لمراجعة قيم متغيرات البرنامج لنرى تصرف البرنامج عند هذا الأمر. في هذه الحالة نضع نقطة توقف break point عند هذا الأمر. في هذه الحالة بالضرب على زر التنفيذ رقم ١ في شكل (٢،٤) سيتم تنفيذ البرنامج حتى هذا الأمر ويتوقف التنفيذ. نقطة التوقف يمكن وضعها أمام أي أمر بوضع علامة الكتابة cursor عند هذا الأمر ثم

الضرب على الأيقونة رقم ٢ في شكل (٢،٤) ستظهر نقطة حمراء مميزة أمام الأمر فورا وبجانب رقمه كما في شكل (٢،٤) حيث تم وضع نقطة توقف أمام الأمر رقم ١.

٤- يمكن إزالة نقطة التوقف عند أي أمر بالنقر على الزر رقم ٣ في شكل (٢،٤) أو حتى بالنقر على النقطة الحمراء نفسها.

٥- يمكن وضع أكثر من نقطة توقف عند أكثر من أمر لتنفيذ البرنامج على أجزاء، حيث بالضرب على زر التنفيذ رقم ١ يتم التنفيذ حتى نقطة التوقف التالية.

٦- لاحظ أن الأمر الذي تقف عنده عملية التنفيذ نتيجة وجود نقطة توقف، يتم وضع سهم أخضر يشير ناحية اليمين بجانب نقطة التوقف التي وصلت إليها عملية التنفيذ.

٧- عند توقف عملية التنفيذ عند أمر معين يشار إليه بالسهم الأخضر يمكن تنفيذ الأوامر بنظام الخطوة بخطوة بحيث مع كل نقرة على المفتاح رقم ٤ في شكل (٢،٤) يتم تنفيذ خطوة واحدة فقط من خطوات البرنامج. تعد هذه الخاصية مهمة جدا في اكتشاف الأخطاء بالذات في البرامج الطويلة، حيث يمكن تنفيذ أمر واحد والتوقف لرؤية تأثير هذا الأمر على النتيجة وهل هي النتيجة المتوقعة أم لا؟.

٨- هناك مجموعة أخرى من أزرار التحكم في تصحيح البرامج، وهي الأزرار ٥ و ٦ و ٧ و ٨ يمكنك تجربتها مباشرة مع أي برنامج لرؤية تأثير كل منها.

٩- يمكنك طلب المساعدة التفصيلية في كل ذلك من قائمة المساعدة Help والدخول على MATLAB ثم Desktop tools and development environment ثم الدخول في Editing and debugging M-Files حيث يمكنك الاطلاع على المزيد من المعلومات عن ملفات الإم.

١٠- هناك الأمر pause الذي يوقف عملية التنفيذ إلى أن تضرب أي زر من أزرار لوحة المفاتيح حيث بعدها تستأنف عملية التنفيذ. نفس برنامج حساب الجذور أعدنا كتابته مرة ثانية بعد إضافة الأمر disp ('..') والذي يعرض أي رسالة بين علامتي التنصيص والأمر pause كما يلي :

```
% Finding roots of a second order equation
a=1;
b=5;
c=6;
disp('Hit any key to continue');
pause
x1=-b/(2*a)+sqrt(b^2-4*a*c)/(2*a)
x2=-b/(2*a)-sqrt(b^2-4*a*c)/(2*a)
```

الآن من مساحة العمل سننفذ البرنامج بكتابة اسم البرنامج example2_1 وهذا هو ما حدث في مساحة العمل :

```
>> example2_1
Hit any key to continue


x1 =
   -2
x2 =
   -3
```

١١- يمكن إضافة رسائل معينة تحذر من أخطاء معينة قد تحدث في البرنامج، فمثلا في برنامج حساب الجذور السابق لو أننا وضعنا a=0 ستظهر النتيجة التالية:

```
>> example2_1
write the value of a: 0
write the value of b: 5
write the value of c: 6
x1 =
  NaN
x2 =
 -Inf
```

بوضع a=0 كان الجذر الأول x1=NaN بمعنى لا يمثل رقم (Not a Number) و x2=-Inf أي سالب مالانهاية، وكل منهما نتائج متوقعة نتيجة القسمة على صفر. لنفترض أننا نريد في حالة إدخال a=0 يتوقف تنفيذ البرنامج ويعطي رسالة معينة. في هذه الحالة يمكن تعديل البرنامج كما يلي:

```
% Finding roots of a second order equation
a=input('write the value of a: ');
if a==0
    error('a must not equal to zero');
end
b=input('write the value of b: ');
c=input('write the value of c: ');
x1=-b/(2*a)+sqrt(b^2-4*a*c)/(2*a)
x2=-b/(2*a)-sqrt(b^2-4*a*c)/(2*a)
```

الجديد في هذا البرنامج هو الأمر if الذي يسأل إذا كانت a=0 بحيث إذا كانت فعلا تساوي صفراً سينفذ الأمر error('...') الذي يطبع الرسالة التي بين القوسين، ثم الأمر end وهو يعني نهاية الأمر الشرطي if وبعد ذلك يخرج ماتلاب من البرنامج دون إكماله ويكتب لك نفس الرسالة كما يلي:

```
>> example2_1
write the value of a: 0
??? Error using ==> example2_1 at 4
a must not equal to zero
```

حيث نلاحظ عدم إكمال البرنامج بمجرد وضع a=0 وكتب لك الرسالة خطأ في البرنامج example2_1 عند السطر ٤، ثم كتب لك الرسالة الموجودة في الأمر error('..').

١٢- أحيانا لا نريد ماتلاب أن يخرج من البرنامج في حالة حدوث مثل هذا الشرط، ولكننا نريد فقط أن يحذرنا ماتلاب من هذا الشرط ويكمل البرنامج حتى آخره. سنعدل البرنامج كما يلي:

```
% Finding roots of a second order equation
a=input('write the value of a: ');
if a==0
   warning('a must not equal to zero, but any way we
     will continue');
end
b=input('write the value of b: ');
c=input('write the value of c: ');
x1=-b/(2*a)+sqrt(b^2-4*a*c)/(2*a)
x2=-b/(2*a)-sqrt(b^2-4*a*c)/(2*a)
```

لاحظ تغيير الأمر error ('..') إلى warning ('..') أي تحذير، ونتيجة تنفيذ البرنامج كانت كالتالي :

```
>> example2_1
write the value of a: 0
Warning: a must not equal to zero, but any way we will continue
> In example2_1 at 4
write the value of b: 5
write the value of c: 6
x1 =
  NaN
x2 =
 -Inf
```

(٢,٣) ملفات الإم للدوال الوظيفية

ملفات الإم التي تعاملنا معها في الجزء السابق تسمى ملفات إم نصية script m-files. معاملات أي ملف من هذه الملفات يجب أن تكون موجودة داخل الملف نفسه. مثلا في برنامج حساب الجذور السابق وضعنا في أول البرنامج a=1 و b=5 و c=6 وعند تنفيذ البرنامج سيستخدم هذه الثوابت. الطريقة الثانية كانت عن طريق الأمر input() بحيث يطلب البرنامج من المستخدم أن يدخل قيم لهذه الثوابت وأطلقنا على هذه الطريقة الطريقة التفاعلية.

نقدم هنا نوعاً ثانياً من ملفات الإم وهو ملفات الدوال. في هذا النوع من الملفات نريد إدخال ثوابت البرنامج كمعاملات مع الاسم. فمثلا برنامج حساب الجذور نريد أن ننفذه بهذه الطريقة :

```
eqroots(1,5,6)
```

حيث بهذه الطريقة a تصبح واحداً و b تصبح خمسة و c تصبح ستة، ويتم حساب الجذور وعرض النتيجة. إنها تماما مثل أن تنادي على إحدى دوال ماتلاب للتنفيذ، مثلا عندما نريد حساب جيب تمام أي زاوية نكتب cos(angle). لكي نرى ذلك سنعدل برنامج حساب الجذور الذي كان اسمه example2_1 إلى الوضع التالي :

```
% Finding roots of a second order equation
function [x1,x2]=eqroots(a,b,c);
if a==0
   warning('a must not equal to zero, but any way we
   will continue');
end
x1=-b/(2*a)+sqrt(b^2-4*a*c)/(2*a);
x2=-b/(2*a)-sqrt(b^2-4*a*c)/(2*a);
```

الجديد هنا والجدير بالملاحظة نلخصه في النقاط التالية :

- البرنامج هو نفسه example2_1.m لم يتغير.
- أول أمر في البرنامج هو الأمر function الذي لابد أن يكون أول أمر في البرنامج. هذا الأمر يتكون من ٤ أجزاء :

١- الأمر function يعقبه مسافة على الأقل.

٢- مخرجات البرنامج توضع في صورة مصفوفة [x1,x2] حيث هذه هي النتائج التي سيخرجها البرنامج أو الدالة، ونفصل بين كل منها بفاصلة.

٣- اسم الدالة وهو هنا eqroots وهو أي اسم يختاره المستخدم.

٤- معاملات الدالة أو مدخلاتها وهي في حالتنا هذه الـ a و b و c ولابد أن توضع بين قوسين كما رأينا ونفصل بين كل منها بفاصلة.

٥- ينتهي هذا الأمر بفاصلة منقوطة.

• الآن لا حاجة لكتابة الثوابت a و b و c لأنها ستدخل مع اسم الدالة من مساحة العمل عند التنفيذ.

• باقي البرنامج كما هو لم يتغير. لاحظ أننا وضعنا فاصلة منقوطة بعد الأمرين لحساب كل من x1 و x2 لأن الدالة ستعرض الخرج مباشرة.

• بعد الانتهاء من كل هذه التعديلات يجب تخزين الملف باسم الدالة، ولذلك سنخزن البرنامج باسم eqroots.

الآن سننفذ هذه الدالة بالطريقة الجديدة التالية:

```
>> [y1,y2]=eqroots(1,5,6)
y1 =
   -2
y2 =
   -3
```

لقد كتبنا المصفوفتين y1 و y2 لتوضع فيهما نتيجة التنفيذ وهي جذرا المعادلة، وأما الثوابت واحد وخمسة وستة فوضعت كمعاملات للدالة سيستخدمها البرنامج في حل المعادلتين x1 و x2 داخليا ثم يضع قيمهما في y1 و y2 كما في أمر التنفيذ.

ميزة التعامل مع هذا النوع من الدوال أنه يمكنك التعامل معه من داخل أي برنامج كدالة من دوال ماتلاب سابقة التجهيز. فمثلا، البرنامج التالي مخزن باسم example2_2 وينادى على الدالة ()eqroots مرتين ويعرض نتيجتها:

```
% using the function eqroots from the program example2_2
[y1,y2]=eqroots(10,2,1)
[z1,z2]=eqroots(4,5,6)
```

وتم تنفيذ البرنامج كالتالي :

```
>> example2_2
y1 =
  -0.1000 + 0.3000i
y2 =
  -0.1000 - 0.3000i
z1 =
  -0.6250 + 1.0533i
z2 =
  -0.6250 - 1.0533i
```

لمزيد من التوضيح نقدم مثالا ثانيا على استخدام الدوال ؛ في البرنامج التالي يقوم المستخدم بإدخال قيمة لنصف القطر، ويقوم البرنامج بحساب وطباعة مساحة الدائرة ومحيطها.

```
% To calculate the area and circumference of a circle
r=input('The radius of the circle(r)= ');
A=pi*r*r;
C=2*pi*r;
disp(['The area of the circle = ',num2str(A)])
disp(['The circumference of the circle = ',num2str(C)])
```

الآن من مساحة العمل سننفذ البرنامج بكتابة اسم البرنامج example2_3 :

```
>> example2_3
The radius of the circle(r)= 6
```

ثم ندخل قيمة نصف القطر ولتكن ٦ سم فيقوم البرنامج بحساب وطباعة مساحة الدائرة ومحيطها كما يلي :

The area of the circle = 113.0973

The circumference of the circle = 37.6991

إذا أردنا استخدام الدوال لإدخال ثوابت البرنامج كمعاملات مع الاسم كما يلي :

[area, circumference] = cal_circle(r)

أي أننا ننشئ دالة تأخذ قيمة نصف القطر وتعطي قيمتين إحداهما للمساحة والأخرى هي المحيط ولذلك نعدل البرنامج للصورة التالية مع مراعاة الاسم عن تخزين البرنامج يكون هو نفس اسم الدالة.

```
function [A,C]=cal_circle(r)
% To calculate the area and circumference of a circle
A=pi*r*r;
C=2*pi*r;
```

من مساحة العمل اكتب الأمر التالي:

>> [Area, Circum]= cal_circle(6)

فتكون النتيجة كالتالي:

```
Area =
  113.0973
Circum =
  37.6991
```

ننصح القارئ أن يجرب مجموعة الأوامر التالية وأن يحاول تفسير النتائج:

```
>> [a]=cal_circle(3)
a =
  28.2743
>> [a,b]=cal_circle(3)
a =
  28.2743
b =
  18.8496
>> [a,b,c]=cal_circle(3)
??? Error using ==> cal_circle
Too many output arguments.
>> cal_circle(3)
ans =
  28.2743
```

للمزيد من التوضيح نقدم فيما يلي مجموعة من التمارين المحلولة كتطبيق على ما تم شرحه في هذا الفصل.

تمارين محلولة

٢-١ اكتب برنامجا لحساب قيمة M و N : $M=\frac{7+x}{z}+\frac{y}{4.5w}$ و $N=\frac{4(x+3y)^2}{z+w}$

```
% Solution of exercise 2-1
% Finding the values of a and b
x=input('write the value of x: ');
```

```
y=input('write the value of y: ');
z=input('write the value of z: ');
if z==0
    warning('z must not equal to zero, but any way we will continue');
end
w=input('write the value of w: ');
if w==0
    warning('w must not equal to zero, but any way we will continue');
end
M=((7+x)/z)+(y/(4.5*w))
N=(4*(x+3*y)^2)/(z+w)
```

٢-٢ اكتب برنامجا لحساب فرق الجهد بين طرفي مقاومة قيمتها ١٠٠٠ أوم إذا كان التيار المار فيها شدته ٥ أمبير.

```
% Solution of exercise 2-2
% Finding the voltage difference
R=1000;
I=5;
V=I*R;
disp(['The voltage difference = I*R = ',num2str(V),' volts'])
```

٢-٣ اكتب برنامجا لحساب قيمة مقاومة مصنوعة من الكربون طولها ٠,٠٢ م ومساحة مقطعها ٧×١٠^-٦ م^٢ والمقاومة النوعية للكربون ٣,٥×١٠^-٥ أوم.متر. إذا كانت : $R=\frac{\rho L}{A}$

```
% Solution of exercise 2-3
% Finding the value of a resistor
L=0.02;
```

```
A=7*10^(-6);
Rho=3.5*10^(-5);
R=(Rho*L)/A
disp(['The value of R = (Rho*L)/A = ',num2str(R),'
ohms'])
```

٢-٤ اكتب برنامجا لحساب المقاومة المكافئة لثلاث مقاومات متصلة على التوالي.

```
% Solution of exercise 2-4
% Finding the equivalent resistance for series resistors
R1, R2, R3
R1=input('write the value of R1: ');
R2=input('write the value of R2: ');
R3=input('write the value of R3: ');
R_equivalent = R1 + R2 + R3;
disp(['The equivalent resistance = R1 + R2 + R3=
                    ',num2str(R_equivalent),' ohms'])
```

٢-٥ اكتب برنامجاً يحول درجة الحرارة من الفهرنهايت Tf إلى الدرجات المئوية Tc والعكس إذا كانت العلاقة بينهما : $T_c = \frac{5}{9}(T_f - 32)$.

```
% Solution of exercise 2-5
% Program to convert temperature from Fahrenheit to
Celsius
Tf=input('Enter the temperature in Fahrenheit: ');
Tc=(5/9)*(Tf-32);
disp(['The temperature = ',num2str(Tc),' degree
celsius'])
```

٢-٦ اكتب برنامجا لحساب مساحة مستطيل ومحيطه:

١- إذا كان الطول يساوي ٦ سم والعرض يساوي ٤ سم.

```
% Solution of exercise 2-6-1
% To calculate the area and perimeter of a rectangle
L=6;
W=4;
A=L*W;
P=(L+W)*2
disp(['The area of the rectangle = ',num2str(A)])
disp(['The perimeter of the rectangle = ',num2str(C)])
```

٢- البرنامج يسأل المستخدم أن يقوم بإدخال قيم الطول والعرض.

```
% Solution of exercise 2-6-2
L=input('The length of the rectangle= ');
W=input('The width of the rectangle= ');
A=L*W;
P=(L+W)*2;
disp(['The area of the rectangle = ',num2str(A)])
disp(['The perimeter of the rectangle = ',num2str(P)])
```

٣- المتغيرات (الطول والعرض) تكون كمدخلات لدالة.

```
function [Area,Perimeter]=cal_rect(L, W)
% To calculate the area and perimeter of a rectangle
Area= L*W;
Perimeter=(L+W)*2;
```

ثم من مساحة العمل اكتب الأمر التالي :

>> [Area, perimeter]= cal_rect(6,4)

٢-٧ اكتب برنامجا لحساب مساحة مثلث إذا كانت العلاقة بين مساحته وأطوال أضلاعه هي : $Area = \sqrt{s(s-L1)(s-L2)(s-L3)}$ و $s=(L1+L2+L3)/2$

```
% Solution of exercise 2-7
% Program to calculate the area of a triangle
L1=input('Enter the first side length: ');
L2=input('Enter the second side length: ');
L3=input('Enter the thirs side length: ');
S=(L1 + L2 + L3)/2;
A=sqrt(S*(S-L1)*(S-L2)*(S-L3))
disp(['The Area of the triangle = ',num2str(A)])
```

نكتفي بهذا القدر عن ملفات الإم M-files حيث يمكن للمستخدم الآن أن يستخدمها في كتابة الكثير من برامجه التطبيقية أو الخوارزمات الخاصة به.

الفصل الثالث

أساسيات استخدام الماتلاب كلغة برمجة عامة

(٣,١) مقدمة

يعد ماتلاب لغة عامة مثله في ذلك مثل أي واحدة من لغات البرمجة ذات المستوى العالي مثل لغة C وC++ وC# و Fortran وغيرها. في هذا الفصل نتعلم أساسيات استخدام الماتلاب كلغة برمجة، ولذلك سنبدأ بالتعرف على المتغيرات والمتجهات والعمليات الحسابية والمنطقية عليها. ثم كيفية التعامل مع البيانات وطرق إدخالها وإخراجها، وبعد ذلك نتناول شرح الحلقات والأوامر الشرطية لفهم كيفية تكرار تنفيذ مجموعة من الأوامر عدداً محدداً أو غير محدد من المرات. لقد أشرنا في الفصل الأول إلى الأرقام المركبة (Complex Numbers)؛ ولذلك سنخصص الجزء الأخير من هذا الفصل لشرح كيفية تعامل الماتلاب مع الأرقام المركبة.

(٣,٢) المتغيرات

المتغيرات في لغة ماتلاب مثل أي لغة برمجة أخرى يجب أن تحقق الشرطين التاليين :

١- تتكون من الحروف الهجائية a حتى z والأرقام من ٠ حتى ٩ والشرطة السفلى underscore (_).

٢- يجب أن يبدأ اسم أي متغير بأحد الحروف الهجائية ، فلا يبدأ برقم أو بحرف من الحروف الخاصة مثل النجمة * أو & أو # وغيرها.

أما طول المتغير فمن الممكن أن يصل إلى أي طول يريده المبرمج ، ولكن ماتلاب يتذكر فقط أول ٦٣ حرفاً من أحرف المتغير. الأمر التالي يعرض لك أقصى طول لأي متغير يعده ماتلاب :

```
>> N = namelengthmax
N =
   63
```

من أمثلة المتغيرات الخاطئة أو التي لا يعدها ماتلاب المتغير student-name لوجود علامة الطرح في وسطه ، والمتغير 2x لأنه يبدأ برقم ، والمتغير $name لوجود علامة الدولار كأحد الأحرف الخاصة. يمكنك أن تسأل ماتلاب عما إذا كان المتغير الذي ستستخدمه يعد متغيرا صحيحا أم لا عن طريق الأمر isvarname التالي الذي يعطيك صفرا إذا كان المتغير غير صحيح أو واحداً إذا كان المتغير صحيحا. انظر إلى المتغيرات التالية لترى أيها يكون مرفوضا من ماتلاب ولماذا وأيها يكون صحيحا :

```
>> isvarname 3cats
ans =
    0
>> isvarname mohamed_eladawy
ans =
    1
>> isvarname mohamed-eladawy
ans =
    0
```

يتم توليد المتغير في الذاكرة بمجرد تعيين قيمة له كما يلي :

```
>> a=5
a =
    5
```

حيث يتم فورا تعيين مكان في الذاكرة وتخزين القيمة أو الثابت ٥ في هذا المتغير. عند التعامل مع أحد المتغيرات غير المعروف مسبقا في ماتلاب، فإن ماتلاب يعطي رسالة خطأ كما يلي :

```
>> a=b*2
??? Undefined function or variable 'b'.
```

في هذا المثال أردنا أن نجعل المتغير a يساوي ضعف المتغير b ولكن لتنفيذ ذلك فإن ماتلاب حاول البحث عن المتغير b فلم يتعرف عليه فأعطى الرسالة التي تفيد أن المتغير b غير معرفٍ عنده، وهذه الرسالة كما نرى مسبوقة بثلاث علامات استفهام.

المتغيرات المحلية local والمتغيرات العامة أو العالمية global

المتغير الذي يتم تعريفه داخل دالة معينة أو داخل برنامج إم معين أو حتى في مجال نافذة الأوامر command window يكون متغيرا محليا local داخل هذه الدالة أو البرنامج أو مجال العمل. بمعنى أنك يمكنك تعريف المتغير a في نافذة الأوامر، وهذا المتغير لن يكون له أي تأثير على متغير a بنفس الاسم داخل دالة أخرى أو برنامج فهذا المتغير محليا تتم رؤيته أو التعامل معه من داخل الكيان الذي تم تعريفه فيه. معظم تعاملاتنا تكون مع المتغيرات المحلية. أما المتغيرات العالمية global أو العامة فهذه المتغيرات يمكن التعامل معها من أكثر من كيان ويتم تعريفها على أنها متغيرات عالمية كما سنرى فيما بعد عند التعامل مع هذا النوع من المتغيرات.

المتغيرات في ماتلاب تختلف عن نظيرتها في اللغات الأخرى

إن ماتلاب ينظر لأي متغير نظرة مختلفة عن نظيرتها في جميع لغات البرمجة الأخرى. إن أي متغير في ماتلاب يعد مصفوفة ، حتى المتغير a=5 أحادي القيمة في المثال السابق ينظر إليه ماتلاب على أنه مصفوفة مكونة من صف واحد وعمود واحد كما أن المتغير :

```
>> b=[ 2  2];
```

عبارة عن مصفوفة من صف واحد وعمودين. أي أن جميع المتغيرات في ماتلاب هي مصفوفات ، حتى المتغير أحادي القيمة يعد مصفوفة من صف واحد وعمود واحد.

حساسية ماتلاب لشكل الحرف

الماتلاب كلغة برمجة حساس لشكل الحرف الأبجدي ، بمعنى أن الأحرف الكبيرة تكون مختلفة عن الأحرف الصغيرة فمثلا المتغير students يختلف عن المتغير STUDENTS وكلاهما يختلف عن المتغير Students فكل واحد من هذه المتغيرات يعد متغيرا قائما بذاته ومختلفا عن المتغيرات الأخرى. هذا على العكس من لغة الباسيك مثلا التي تعد كل هذه المتغيرات كمتغير واحد.

إن جميع أوامر ماتلاب المبنية بداخله كلها يتم النداء عليها عن طريق كتابتها بالأحرف الصغيرة فقط لأن النداء عليها بالأحرف الكبيرة سيعطي رسالة خطأ كما في الأمر clc والذي يعني تنظيف الشاشة أو إخلاءها من أي كتابة فيها. كتابة هذا الأمر بالأحرف الكبيرة لن يتعرف عليه ماتلاب كما في الأمر التالي :

```
>> CLC
??? Undefined function or variable 'CLC'.
```

بينما كتابة نفس الأمر بالأحرف الصغيرة كما يلي فسيتم التعرف عليه .

```
>> clc
```

(٣،٣) المتجهات

كما ذكرنا مسبقا، فإن أساس المتغيرات في ماتلاب هو المصفوفة، حتى أن الكمية الثابتة الأحادية تعد في ماتلاب مصفوفة ذات صف واحد وعمود واحد. المصفوفة هي كيان حسابي له عدد من الصفوف وعدد من الأعمدة، بينما المتجه فهو كيان له صف واحد أو عمود واحد. انظر إلى الأمر التالي :

```
>> a=[1 2 3 4 5];
```

والذي يعرف المتغير a كمتجه من خمسة عناصر من صف واحد. لذلك يمكن أن نطلب من ماتلاب أن يعرض هذا المتغير باستخدام الأمر disp() كما يلي :

```
>> disp(a)
   1    2    3    4    5
```

باستخدام الأمر whos يمكن أن نعرف بعض المعلومات عن المتغير a كما يلي :

```
>> whos
  Name      Size            Bytes  Class     Attributes

  a         1x5                40  double
```

حيث نلاحظ أن المتغير الذي اسمه a هو مصفوفة من صف واحد وخمسة أعمدة (أي أنه عبارة عن متجه) وهذا المتجه يتكون من خمسة عناصر تشغل ٤٠ بايتاً، بمعنى أن كل عنصر يتم تمثيله في ٨ بايت، وهو ما يعبر عنه بالتمثيل متضاعف الدقة double precession. يمكن استخدام الفاصلة للفصل بين عناصر المتجه كما يلي :

```
>> a=[1,2,2,4,5]
a =
   1    2    2    4    5
```

كما أن المسافة أو الفاصلة تستخدم للفصل بين عناصر الصف الواحد، فإن الفاصلة المنقوطة تستخدم للفصل بين الأعمدة. في المثال التالي المتغير b يتكون من خمسة صفوف وعمود واحد ولذلك يتم إدخاله كالتالي :

```
>> b=[1;2;3;4;5];
```

ولذلك فعند عرض المتغير b سيظهر كما يلي :

```
>> disp(b)
    1
    2
    3
    4
    5
```

حيث كما نرى أن ماتلاب قد مثل المتغير b في خمسة صفوف وعمود واحد.
يمكن استخدام المتجه كعنصر في متجه آخر كما يلي ، حيث تم استخدام المتجهين a و b لتكوين المتجه c :

```
>> a=[1 2 3];
>> b=[4 5];
>> c=[a b];
>> disp(c)
    1    2    3    4    5
```

انظر لهذا المثال :

```
>> a=[1;2;3];
>> b=[4;5;6];
>> c=[a b];
>> disp(c)
c =
    1    4
    2    5
    3    6
```

يمكن الإعلان عن متغير فارغ كما يلي :

```
>> x=[ ];
>> disp(x)
>>
```

حيث عند محاولة عرض المتغير x لم يتم عرض أي شيء ، مما يعنى أن المتغير x فارغ أو لا يحتوي أي شيء. يجب أن نركز هنا على أن المتغير x في هذه الحالة لا يساوي صفراً لأن الصفر يعنى أن x عبارة عن مصفوفة من صف واحد وعمود واحد له القيمة صفر ، أما المتغير الفارغ فلا يحتوي أي شيء حتى القيمة صفر.

يمكن تخصيص قيم ابتدائية للمتغيرات بأكثر من طريقة كما يلي :

```
>> x=1:5;
>> disp(x)
   1   2   3   4   5
```

هنا تم استخدام النقطتين (الكولون) في تحديد قيم ابتدائية للمتغير x حيث ستأخذ x القيم من واحد حتى خمسة بفارق واحد بين كل عنصر والتالي له. هنا سنجعل المتغير x يأخذ القيم من واحد حتى ثلاثة بفارق ٠.٥ بين كل قيمة والتالية لها.

```
>> x=1:0.5:3;
>> disp(x)
   1.0000   1.5000   2.0000   2.5000   3.0000
```

لاحظ أن الخطوة أو الفرق بين كل قيمة أو عنصر والتالي له توضع بين القيمة الابتدائية والقيمة النهائية للمتغير ويفصل بين كل منها النقطتان أو الكولون. يمكن للخطوة أن تكون سالبة لتوليد متجه من القيم المتناقصة.

الدالة linspace يمكن استخدامها أيضا لوضع قيما ابتدائية للمتجهات. الدالة linspace (x1,x2,N) تضع قيما ابتدائية لمتجه يتكون من N من القيم المتساوية بين القيمة x1 والقيمة x2. من أمثلة ذلك ما يلي :

```
>> linspace(0,10,5)
ans =
      0   2.5000   5.0000   7.5000   10.0000
>> linspace(0,pi/2,5)
ans =
      0   0.3927   0.7854   1.1781   1.5708
```

الدالة linspace (x1,x2) تضع متجها من ١٠٠ قيمة متساوية بين القيمة x1 و x2.

عملية تحويل الأعمدة إلى صفوف والصفوف إلى أعمدة لأي مصفوفة تسمى transpose وعند تطبيق ذلك على متجه الصف الواحد، فإنه يتحول إلى متجه العمود

الواحد والعكس. عملية التحويل هذه يرمز لها بالرمز (') وهي علامة التنصيص الأحادية. انظر إلى الأمثلة التالية :

```
>> x=[1 2 3 4 5]
x =
    1    2    3    4    5
```

حيث تم تحديد المتجه x المكون من صف واحد وخمسة أعمدة.

```
>> y=x'
y =
    1
    2
    3
    4
    5
```

حيث تم تحديد المتجه y الذي يساوي المتجه x بعد دورانه بحيث أصبح المتجه y مكونا من خمسة صفوف وعمود واحد.

```
>> z=y'
z =
    1    2    3    4    5
```

الآن حددنا المتجه z الذي يساوي المتجه y بعد دورانه بحيث أصبح يساوي المتجه x الأصلي حيث أصبح مكونا من صف واحد وخمسة أعمدة مرة ثانية.

الآن سنحدد متجهاً من سبعة عناصر بقيم عشوائية أقل من الواحد :

```
>> r=rand(1,7)
r =
   0.8147   0.9058   0.1270   0.9134   0.6324   0.0975   0.2785
```

سنعرض العنصر رقم ٣ كما في الأمر التالي :

```
>> r(3)
ans =
   0.1270
```

بل يمكن أن نعرض العناصر من الثاني حتى الخامس كما يلي :

```
>> r(2:5)
ans =
   0.9058   0.1270   0.9134   0.6324
```

بل يمكن عرض العناصر من ١ حتى ٧ بخطوة مقدارها ٢ كما يلي :

```
>> r(1:2:7)
ans =
   0.8147    0.1270    0.6324    0.2785
```

أيضا يمكن عرض متجه مكون من عناصر متفرقة من المتجه r كما يلي حيث سنعرض متجهاً مكوناً من العناصر الثاني والسابع والخامس والرابع :

```
>> r([2 7 5 4])
ans =
   0.9058    0.2785    0.6324    0.9134
```

يمكن تفريغ أي عدد من عناصر المصفوفة r بجعله عنصرا فارغا كما يلي :

```
>> disp(r)
   0.8147    0.9058    0.1270    0.9134    0.6324    0.0975    0.2785
>> r([2 4 5 7])=[ ]
r =
   0.8147    0.1270    0.0975
```

حيث نلاحظ عدم وجود العناصر الثاني والرابع والخامس والسابع في المتجه r.

نلاحظ مما سبق أن ترقيم عناصر المتجه (وكذلك المصفوفة) في ماتلاب يبدأ من العنصر رقم ١، على العكس من بعض لغات البرمجة الأخرى التي تبدأ الترقيم من الصفر، أي أن العنصر الأول في ماتلاب يكون العنصر رقم واحد.

(٣،٤) المصفوفات

يمكن اعتبار المصفوفة جدولا مكونا من عدد من الصفوف والأعمدة. يمكن عرض مصفوفة مكونة من صفين وثلاثة أعمدة كما يلي، حيث تستخدم الفاصلة المنقوطة للدلالة على نهاية عمود :

```
>> a=[1 2 3;4 5 6]
a =
    1    2    3
    4    5    6
```

يمكن تدوير المصفوفة a بجعل الصفوف أعمدة والأعمدة صفوف كما يلي :

```
>> a'
ans =
   1    4
   2    5
   3    6
```

يمكن تكوين المصفوفة من عدد من متجهات الأعمدة ، حيث سنكون مصفوفة اسمها table المكونة من العمود الأول الذي يمثل الزوايا من صفر حتى ١٨٠ درجة بخطوة مقدارها ٣٠ درجة والعمود الثاني هو جيب تمام هذه الزوايا كما يلي :

```
>> x=0:30:180;
>> table=[x' sin(x*pi/180)']
table =
        0         0
  30.0000    0.5000
  60.0000    0.8660
  90.0000    1.0000
 120.0000    0.8660
 150.0000    0.5000
 180.0000    0.0000
```

(٣,٥) العمليات والتعبيرات في ماتلاب

جميع لغات البرمجة وبالتالي جميع البرامج لابد أن تتعامل مع العمليات الحسابية والمنطقية وكذلك مع التعبيرات الحسابية والمنطقية أيضا. فما هو الموقف بالنسبة لهذه العمليات في الماتلاب وبالذات في ظل التعامل مع المتغيرات المصفوفية والمتجهية التي هي أساس التعامل في ماتلاب؟. يتكون أي تعبير من مجموعة من الأرقام أو الثوابت والمتغيرات والتي تُجرى عليها بعض العمليات مثل التعبير a*2+0.003*b.

ننصح المستخدم أن يحاول كتابة الأوامر التالية لمعرفة الحروف الخاصة والتعبيرات الموجودة في الماتلاب، والتي تستخدم في العمليات الحسابية والمنطقية :

```
help ops <<
help relop <<
help arith<<
help slash<<
```

بالنسبة للأرقام في ماتلاب يتم التعبير عنها في الصورة الحقيقية ذات العلامة العشرية أو بدونها كما يلي :

```
>> 1.234
ans =
   1.2340
>> 456
ans =
  456
>> 0.0092
ans =
   0.0092
```

فكل هذه صور لأرقام يمكن أن نتعامل بها خلال الماتلاب. هناك أيضا الصورة العلمية أو الصورة الأسية، والتي تستخدم للتعبير عن الأرقام المتناهية الكبر أو الصغر مثل $1.234x10^9$ والتي يتم التعبير عنها كما يلي :

```
>> 1.234e9
ans =
 1.2340e+009
```

أو الأرقام الصغيرة مثل :

```
>> 1.234e-20
ans =
 1.2340e-020
```

لاحظ أن :

```
>> 1.234*10^9
ans =
 1.2340e+009
```

يعتبر تعبيرا في ماتلاب يحسب نفس الرقم 1.234e+009 ولكن عن طريق عملية ضرب للرقم 1.234 في القيمة الأسية 10^{+9} وهذه تعتبر عملية حسابية تأخذ الكثير من الوقت على العكس من الكمية 1.234e+009 التي تعتبر رقما أو ثابتا يتم فقط حجز الذاكرة اللازمة له. إن أصغر رقم حقيقي وأكبر رقم حقيقي يمكن التعامل معهما في ماتلاب يمكن بيانهما باستخدام الأمرين التاليين :

```
>> realmin
ans =
 2.2251e-308
```

والذي يبين أن أصغر رقم حقيقي هو ٢,٢٢٥١*١٠^{-٣٠٨}.

```
>> realmax
ans =
 1.7977e+308
```

الذي يبين أن أكبر رقم حقيقي هو ١,٧٩٧٧*١٠^{+٣٠٨}.

وأما الدقة precision التي يتعامل بها ماتلاب، والتي تمثل أصغر فارق بين أي رقمين يمكن أن يتعامل معهما ماتلاب فيمكن إظهارها بالأمر التالي :

```
>> eps
ans =
 2.2204e-016
```

والتي تعتبر كمية صغيرة جدا تصل إلى ١٦ خانة على يمين العلامة العشرية. بالنسبة لطرق تمثيل هذه البيانات في ماتلاب سيأتي ذكرها في مواضع مختلفة لاحقا.

العمليات الحسابية في ماتلاب

جدول (٣.١) يبين العمليات الحسابية التي يتعامل معها الماتلاب، حيث نلاحظ أن عمليات الجمع والطرح والقسمة اليمنى هي نفسها في الصورة الجبرية والصورة المستخدمة في ماتلاب. أما الرمز المستخدم في ماتلاب لعملية الضرب فهو النجمة * بدلا من × . الجديد أيضا هو عملية القسمة العكسية أو اليسرى، حيث

تستخدم الشرطة المائلة ناحية اليسار لقسمة الرقم الذي على يمينها على الرقم الذي على يسارها ، أما عملية الأس فهي العلامة ^ .

جدول (٣.١). العمليات الحسابية في ماتلاب.

العملية	الصورة الجبرية	الصورة المستخدمة في ماتلاب
الجمع	a+b	a+b
الطرح	a-b	a-b
الضرب	axb	a*b
القسمة اليمنى	a/b	a/b
القسمة اليسرى	b/a	a\b
الأس	a^b	a^b

من المهم جدا أن نعرف ترتيب تنفيذ العمليات الحسابية في حالة وجود أكثر من عملية حسابية في نفس التعبير. فمثلا في التعبير g*t^2 هل سيقوم ماتلاب بضرب g في t ثم يرفع الناتج للأس 2، أم سيرفع t للأس 2 أولا ثم يضرب الناتج في g ؟ إن ما يجيب على ذلك هو طريقة ماتلاب التي سيتبعها في أسبقية العمليات precedence of operations. الجدول (٣.٢) يبين ترتيب أسبقية هذه العمليات. نلاحظ من هذا الجدول أن أول ما يبدأ به ماتلاب هو حساب ما بداخل الأقواس. بعد ذلك أول ما يتم تنفيذه هو الأسس من اليسار لليمين، ثم الضرب والقسمة من اليسار لليمين، ثم أخيرا الجمع والطرح من اليسار لليمين. المقصود من اليسار لليمين هنا هو أنه في حالة وجود عمليتين لهما نفس الأسبقية في نفس التعبير، فإن ماتلاب سينفذ أول عملية منهما تأتى من ناحية اليسار. فمثلا في التعبير a/b*c سيبدأ من ناحية اليسار بقسمة a على b ثم يضرب ناتج القسمة في c. أي أنه بفرض أن a=9 و b=3 و c=2 فإن ناتج هذا التعبير سيكون ٦. أما بالنسبة للتعبير a+b*c فإن متلاب سيبدأ من اليسار لينفذ العملية ذات الأسبقية الأعلى، وهي ضرب b في c ثم يجمع الناتج مع a لأن عملية الجمع لها أولوية أقل من عملية الضرب. وعلى ذلك فإنه باستخدام نفس القيم السابقة، فإن ناتج هذا التعبير سيكون ١٥.

جدول (٣،٢) أسبقية العمليات الحسابية في ماتلاب

الأولوية	رمز العملية في ماتلاب	العملية (من اليسار إلى اليمين)
١	( )	الأقواس المستديرة.
٢	^، .^، '	الأس، الأس على كل عنصر، الدوران.
٣	+ (unary plus), - (unary minus), ~ (NOT)	جعل عنصر موجب، جعل عنصر سالب، العكس.
٤	*، /، \، .*، ./، .\	الضرب، القسمة اليمنى، القسمة اليسرى، الضرب في جميع العناصر، قسمة جميع العناصر يمينيا، قسمة جميع العناصر يساريا.
٥	+، −	الجمع والطرح.
٦	:	مجموعة من العناصر
٧	<، >، <=، >=، ==، ~=	أكبر من، أصغر من، أكبر من أو يساوى، أصغر من أو يساوى، يساوى، لا يساوى.
٨	& (AND)	عملية الآند.
٩		(OR)	عملية الأور.

يمكن إجراء عملية معينة على مجموعة من عناصر المتجه الواحد كما في المثال التالي :

```
>> a=[1 2 3 4 5]
a =
   1    2    3    4    5
>> 1 + a(1:5)
ans =
   2    3    4    5    6
```

حيث تم تحديد المتجه a من خمسة عناصر كما في الأمر الأول ، ثم تم إضافة واحد لجميع العناصر من الأول حتى الخامس ، وهو العملية (٥ : ١) إذا أسميناها عملية.

يمكن إجراء عمليات على عناصر المتجهات المتساوية الطول. فمثلا لنفترض المتجهين a و b كما يلي :

```
>> a=[1 2 3 4 5];
>> b=[6 7 8 9 10];
```

الآن سنضرب كل عنصر في المتجه a فيما يقابله من عناصر المتجه b كما يلي :

```
>> c=a.*b
c =
   6   14   24   36   50
```

ثم سنقسم كل عنصر من عناصر المتجه a على ما يقابله من عناصر المتجه b قسمة يمينية كما يلي :

```
>> d=a./b
d =
   0.1667   0.2857   0.3750   0.4444   0.5000
```

ثم سنقسم كل عنصر من عناصر المتجه b على ما يقابله من عناصر المتجه a قسمة يسارية كما يلي :

```
>> x=a.\b
x =
   6.0000   3.5000   2.6667   2.2500   2.0000
```

يمكن أن نرفع جميع عناصر المتجه إلى الأس ٢ كما يلي :

```
>> a=[1 2 3];
>> c=a.^2
c =
   1   4   9
```

كما يمكن رفع عناصر عنصر معين إلى الأس الموجود في العنصر المقابل له في متجه آخر كما يلي :

```
>> a=[1 2 3];
>> b=[2 3 4];
>> a.^b
ans =
   1   8   81
```

يمكن ضرب أو قسمة كل عناصر المتجه على قيمة واحدة ثابتة باستخدام عمليات الضرب والقسمة العادية بدون استخدام النقطة كما يلي :

```
>> a=[1 2 3];
>> c=a*2
c =
   2    4    6
>> c=a/2
c =
   0.5000    1.0000    1.5000
```

لجمع أو طرح كل عنصر في متجه معين مع ما يقابله من عناصر متجه آخر نستخدم الجمع أو الطرح العادي كما يلي :

```
>> c=a+b
c =
   3    5    7
>> d=a-b
d =
  -1   -1   -1
```

نلاحظ من ذلك أن النقطة تكون ضرورية في حالات ضرب أو قسمة أو حساب الأس لكل عنصر في مصفوفة مع ما يقابله من عناصر المصفوفة الأخرى، بينما هذه النقطة تكون غير ضرورية في حالتي الجمع والطرح.

ونؤكد على أنه في حالة إجراء العمليات الحسابية النقطية على المصفوفات لابد أن تكون هذه المصفوفات لها نفس الحجم.

لاحظ عند تنفيذ الأوامر الآتية أنه في بعض الحالات هناك خطأ، ولم يتم التنفيذ لعدم تساوي حجم المصفوفات:

```
>> a=[1 2 3; 4 5 6]
a =
    1    2    3
    4    5    6

>> b=a'
b =
    1    4
    2    5
    3    6
>> c=[9 8 7; 6 5 4]
c =
    9    8    7
    6    5    4
>> a+b
??? Error using ==> plus
Matrix dimensions must agree.
>> a+c
ans =
   10   10   10
   10   10   10
>> a-b
??? Error using ==> minus
Matrix dimensions must agree.
>> a-c
ans =
   -8   -6   -4
   -2    0    2
>> a.*b
??? Error using ==> times
Matrix dimensions must agree.
>> a*b
ans =
   14   32
   32   77
>> a*c
??? Error using ==> mtimes
Inner matrix dimensions must agree.
>> a.*c
ans =
```

```
   9   16   21
  24   25   24
>> a/b
??? Error using ==> mrdivide
Matrix dimensions must agree.
>> a./b
??? Error using ==> rdivide
Matrix dimensions must agree.

>> a/c
ans =
   2.3333  -3.3333
   3.3333  -4.3333
>> a./c
ans =
   0.1111    0.2500    0.4286
   0.6667    1.0000    1.5000
>> a+b+c
??? Error using ==> plus
Matrix dimensions must agree.
>> a+b'+c
ans =
   11    12    13
   14    15    16
```

(٣,٦) عرض البيانات

يمكن عرض البيانات في ماتلاب بطريقتين، الأولى عن طريق كتابة اسم المتغير المراد عرض قيمته، أو حتى كتابة التعبير المراد عرض قيمته بدون فاصلة منقوطة كما في المثال التالي :

```
>> a=[1 2 3 4 5];
>> a
a =
   1    2    3    4    5
```

بمجرد أن كتبنا المتغير a ثم enter ظهرت قيمة المتغير a.

الطريقة الثانية باستخدام الأمر disp() كما في المثال التالي :

```
>> disp(a)
   1    2    3    4    5
```

(٣,٧) التكرار أو الحلقات باستخدام الأمر for

مثل أي لغة من لغات البرمجة لابد أن تكون هناك وسيلة أو أوامر لتنفيذ مقطع معين من البرنامج أو عدة أوامر منه عدد معين من المرات. انظر لما يلي :

```
>> for i=1:5, disp(i), end
   1
   2
   3
   4
   5
```

حيث نلاحظ في هذا الأمر أن i تتغير من واحد لخمسة وفي كل مرة يتم عرض قيمة المتغير i، مما يعنى تنفيذ الأمر disp (i) خمس مرات. المثال التالي يعتبر مثالا حيا على استخدام الأمر for. البرنامج يحسب الجذر التربيعي لأي رقم باستخدام طريقة نيوتن.

```
%Neuton method to calculate square root of a number a
a=2;
x=a/2;
disp('Using Newton method to calculate square root of
a=2');
for I = 1:6
    x=(x+a/x)/2;
    disp( x )
end
disp('Using the sqrt function');
disp( sqrt(2) );
```

ونتيجة البرنامج هي كما يلي حيث نلاحظ الوصول إلى قيمة الجذر بعد ست محاولات حيث عندها أصبح الجذر المحسوب بهذه الطريقة مساويا للجذر المحسوب باستخدام الدالة الجاهزة في ماتلاب.

```
Using Newton method to calculate square root of a=2
  1.5000
  1.4167
  1.4142
  1.4142
  1.4142
  1.4142
Using the sqrt function
  1.4142
```

المثال التالي أيضا يوضح استخدام الأمر for لحساب مضروب مجموعة من الأرقام حتى رقم معين وفي هذا المثال حتى الرقم ٦.

```
>> n = 6; fact = 1;
  for k = 1:n
  fact=k*fact;
  disp( [k fact] )
end

    1     1
    2     2
    3     6
    4    24
    5   120
    6   720
```

إذا كنت تريد عرض مضروب آخر رقم فقط يمكنك أن تأخذ الأمر disp ([k fact]) خارج جسم الحلقة، أي بعد الأمر end.

إذن، وكما نلاحظ مما سبق، أنه يتم تنفيذ مجموع الأوامر بين الأمر for و end عدداً من المرات مساوياً للقيمة المحددة بواسطة مؤشر الحلقة الذي هو المتغير k في المثال السابق. الشكل العام لهذا النوع من الحلقات يمكن وضعه على الصورة التالية:

```
for index = j:m:k
          statements;
end
```

من هذا الشكل العام يمكننا أن نلاحظ النقاط التالية :

١- مؤشر الحلقة هو متغير (مثل المتغير k في المثال السابق) يتغير من القيمة الابتدائية j حتى القيمة النهائية k وفي كل مرة يزداد المؤشر بمقدار m. أي أن m هي الخطوة التي يزداد بها هذا المؤشر في كل مرة إلى أن يساوي أو يزيد عن القيمة النهائية k حيث عند ذلك يتوقف تنفيذ الحلقة ، وينتقل التنفيذ إلى الأمر التالي.

٢- عند اكتمال تنفيذ الحلقة ، فإن المؤشر يأخذ آخر قيمة يصل إليها.

٣- ما بين الأمر for و end هو ما يسمى بجسم الحلقة وهو يكون أمراً واحداً أو مجموعة من الأوامر يتم تنفيذها في كل مرة تتغير فيها قيمة المؤشر.

٤- لابد من وجود الأمر end لكي يوضح نهاية الحلقة أو نهاية جسم الحلقة.

٥- لا تنسى وضع الفواصل أو الفواصل المنقوطة في نهاية كل أمر.

هل تريد معرفة الزمن الذي يأخذه الحاسب الذي تعمل عليه في تنفيذ برنامجك أو حتى مقطعا من هذا البرنامج؟ إن هذا ممكن بسهولة عن طريق وضع كلمة tic في بداية البرنامج (أو بداية المقطع المراد حساب زمنه) ثم وضع كلمة toc في نهاية البرنامج (أو نهاية المقطع) حيث سيقوم ماتلاب بعرض الزمن المأخوذ في التنفيذ ما بين الـ tic والـ toc بالثانية كما في المثال التالي :

```
%execution time
tic
s=0;
for i=1:10000;
    s=s+1;
end
disp(s);
toc;
```

وستكون النتيجة كالتالي :

```
10000
Elapsed time is 0.000284 seconds.
```

يجب أن نشير أن قيمة الزمن المأخوذ في التنفيذ يختلف من حاسب آلي إلى آخر، ويعتمد على سرعة المعالج المستخدم، وكذلك إذا كان الحاسب ينفذ أكثر من أمر بالتوازي.

تداخل أو تعشيق الحلقات for

الصورة العامة للحلقات for المتداخلة هي كالتالي :

```
for index = j:m:k
   for indexi =p:q:r
        statements;
   end
end
```

حيث عند تنفيذ هذه التشكيلة من الحلقة for سيبدأ التنفيذ بالـ for الخارجية، حيث مؤشر هذه الحلقة الخارجية index يأخذ القيمة الابتدائية j، ثم يدخل على الحلقة for الداخلية، حيث يبدأ مؤشرها indexi بالقيمة p، ثم ينفذ مجموعة الأوامر statements حتى أول end ثم يعود ليزيد المؤشر indexi بمقدار الخطوة q ثم يدخل في تنفيذ جسم الحلقة statements، وهكذا يستمر التنفيذ للحلقة الداخلية مستمرا حتى يصل مؤشرها indexi إلى القيمة النهائية r حيث يتم تنفيذ الحلقة لآخر مرة، ثم يخرج التنفيذ من الحلقة الداخلية ليصطدم بنهاية end الحلقة الخارجية فيرجع مرة أخرى ليزيد مؤشرها index بمقدار m ويدخل للحلقة الداخلية فينفذها بالكامل حتى يصل مؤشرها إلى القيمة النهائية. ويخرج منها ليعاود تنفيذ الحلقة الخارجية مرة أخرى، وهكذا حتى ينتهي من تنفيذ الحلقة الخارجية ويخرج منها لتكملة البرنامج. أي أن جسم الحلقة الداخلية statements سيتم تنفيذه عدداً من المرات يساوي عدد مرات تنفيذ الحلقة الداخلية في عدد مرات تنفيذ الحلقة الخارجية.

يتضح ذلك من المثال التالي الذي يحسب المصفوفة c التي تساوي حاصل جمع المصفوفتين a و b من خلال برنامج، وليس من خلال أمر الماتلاب المباشر لجمع مصفوفتين.

```
% Nested for statements
a=[1 2 3;4 5 6;7 8 9];
b=[1 2 3;4 5 6;7 8 9];
k=0;
for i=1:1:3
    for j=1:1:3
        c(i,j)=a(i,j)+b(i,j);
        k=k+1;
    end
end
disp(c);
disp(['Execution of the inner loop body equals:    '
num2str(k)]);
```

هذا البرنامج يتكون من حلقتين for متداخلتين؛ مؤشر كل منهما يتغير من واحد إلى ثلاثة ولذلك فإن جسم الحلقة الداخلية سينفذ ٣*٣=٩ مرات؛ ولذلك وضعنا العداد k داخل الحلقة الداخلية لنعد عدد المرات التي سينفذ فيها البرنامج الحلقة الداخلية، وتم عرض قيمة k لتكون نتيجة البرنامج كما يلي:

```
    2    4    6
    8   10   12
   14   16   18
Execution of the inner loop body equals:   9
```

يمكن للحلقات for أن تتداخل لأي عمق بلا حدود، فقط يجب الحذر من نهاية كل حلقة يجب أن تكون في مكانها الصحيح. ولحسن الحظ، فإنه أثناء تحرير وكتابة البرنامج فإن ماتلاب يبين لك ذلك عن طريق تنسيق أسطر الكتابة كما في البرنامج السابق.

الحلقة for من الحلقات التي يكون عدد مرات تنفيذها معلوما مقدما من خلال القيمة الابتدائية والقيمة النهائية والخطوة لمؤشر هذه الحلقة. هناك بعض المواقف يكون من الصعب معرفة عدد مرات تنفيذ الحلقة مما يجعل من الصعب استعمال الحلقة for. كمثال على ذلك افترض أننا نصمم برنامج يقرأ درجة الحرارة وينفذ خطوات معينة

طالما أن درجة الحرارة أقل من ٤٠ درجة مئوية مثلا ، ويخرج من البرنامج بمجرد وصول الحرارة إلى ٤٠ درجة. هل يعلم مستخدم البرنامج أو المصمم متى ستصل درجة الحرارة إلى ٤٠ حتى يحسب عدد مرات تنفيذ البرنامج؟ بالطبع لا ؛ لأن الحرارة تزيد وتنقص. لذلك كان لابد من وجود وسيلة أخرى لعمل حلقات لا نعلم عدد مرات تنفيذها مسبقا كما في الحلقة for. هذا النوع من الحلقات سنؤجله بعد دراسة القرارات في الجزء التالي.

(٣.٨) القرارات

الشروط في لغة ماتلاب مثل أي لغة ، حيث يتم اتخاذ قرار معين بناء على نتيجة شرط معين. الصورة العامة لأمر الشرط في ماتلاب هي :

if condition statement, end

حيث يتم اختبار الشرط condition statement فإذا كان حقيقيا سيتم تنفيذ مجموعة من الأوامر وإذا لم يكن الشرط حقيقيا سيتم تنفيذ مجموعة أخرى من الأوامر. انظر مبدئيا إلى المثال التالي :

```
%If statement
course_grade='fail';
course_mark=80;
if (course_mark)>=60
    course_grade='pass';
end
disp(['course_grade =  '   course_grade]);
```

في هذا المثال الشرط هو course_mark>=60 ، فإذا كان الشرط حقيقيا بمعنى أن الـ course_mark فعلا أكبر من أو يساوي ٦٠ كما هو الحال حيث course_mark=80 فإن الشرط سيكون حقيقيا ، وسينفذ ماتلاب أوامر الشرط ، وهي أمر واحد فقط في هذه

الحالة course_grade='pass' وهو الأمر التالي للشرط مباشرة ثم يذهب للتنفيذ بعد الأمر end وهي أمر العرض الذي ستكون نتيجته course_grade=pass ، أما إذا وضعنا المتغير course_mark=40 مثلا ، فإنه في هذه الحالة سيكون الشرط غير حقيقيا ؛ وبالتالي فإن أمر الشرط course_grade=pass لن ينفذ وسينتقل التنفيذ مباشرة لما بعد الأمر end حيث سينفذ أمر العرض ويعرض course_grade=fail المحدد في بداية البرنامج. نتيجة تنفيذ هذا البرنامج هي :

```
course_grade =  pass
```

جميع أوامر الشرط التي تستخدم هنا تكون أوامر منطقية بمعنى أن نتيجة هذه الأوامر تكون حقيقية true أو غير حقيقية false ، ولذلك فإن نتيجة هذه الشروط تكون واحداً إذا كانت حقيقية وتكون صفراً إذا كانت غير حقيقية. ولذلك انظر لهذه الأوامر المنطقية ونتيجة ماتلاب لها :

```
>> 2>0
ans =
    1
```

في هذا الأمر نتيجة الشرط ٢>٠ هي بالطبع نتيجة حقيقية ؛ ولذلك كانت نتيجة ماتلاب لها تساوي واحداً كما رأينا. أما الشرط التالي :

```
>> 2<0
ans =
    0
```

فهو بالطبع شرط غير حقيقي ولذلك كانت نتيجته صفرا كما رأينا. يمكن لهذه الشروط أن تكون شروطا مركبة أو أكثر تعقيدا كما سنرى في الأمثلة التالية.

في مثال حساب التقديرات السابق اعتبرنا أن أي درجة أعلى من أو تساوي ٦٠ تقابل التقدير مقبول pass. الآن افترض أننا نريد أن نحدد التقدير مقبولاً بأن يقابل فقط

الدرجة أكبر من أو تساوي ٦٠ ولكنها أقل من ٧٠. سنعيد كتابة البرنامج السابق مرة ثانية مع تعديل أمر الشرط ليحقق ذلك كما يلي :

```
%If statement
course_grade='fail';
course_mark=40;
if (course_mark >=60 && course_mark < 70)
    course_grade='pass';
end
disp(['course_grade='   course_grade]);
```

المشكلة مع هذا البرنامج أنه سيعطي الدرجة الأكبر من أو تساوي ٦٠ وأقل من ٧٠ التقدير مقبولاً pass ولكن أي درجة خارج هذا المدى سواء أقل أو أكبر منه سيعطيها التقدير راسباً fail وهذا خطأ بالطبع ؛ لذلك سنعدل البرنامج السابق ليغطى كل مدى الدرجات من صفر حتى ١٠٠ ويعطي التقدير المناسب لكل مدى ، وسنجعل البرنامج أكثر تفاعلية من خلال استخدام الأمر input كما يلي :

```
%If statement
course_mark=input('Mark= ');
if (course_mark < 60 )
    course_grade='fail';
end
if (course_mark >=60 && course_mark < 70)
    course_grade='pass';
end
if (course_mark >=70 && course_mark < 80)
    course_grade='Good';
end
if (course_mark >=80 && course_mark < 90)
    course_grade='Very Good';
end
if (course_mark >=90 )
    course_grade='Excellent';
end
disp(['course_grade = '   course_grade]);
```

وهذا خرج البرنامج لعدد من الحالات لاختباره :

Mark = 30
course_grade = fail

Mark = 57
course_grade = fail

Mark = 72
course_grade = Good

Mark = 84
course_grade = Very Good

Mark = 91
course_grade = Excellent

حيث باستخدام الأمر input يتم إدخال الدرجة ويرد البرنامج فورا بالتقدير المناسب.

أحيانا يحتاج الموقف إلى تصنيف قيمة معينة إلى واحدة من حالتين، فمثلا نريد إدخال رقما معينا ويرد علينا البرنامج، هل الرقم الذي تم إدخاله رقما زوجيا أم فرديا؟ هنا من المفضل استخدام الأمر if....else كما في المثال التالي :

```
%Odd even
x=input('write any integer  ');
if rem(x, 2) == 0
    disp('This number is even')
else
    disp('This number is odd');
end
```

ونتيجة اختبار البرنامج ستكون كالتالي :

write any integer 50
This number is even

write any integer 13
This number is odd

الدالة rem(x,y) هي اختصار لكلمة remainder وتحسب باقي قسمة x على y.

في هذا البرنامج الدالة rem ستقوم بقسمة المتغير x على ٢ وتحتفظ بباقي القسمة فقط، حيث الأمر if سيختبر هذا الباقي، فإذا كان الباقي يساوي صفراً فذلك يعنى أن الرقم زوجي وبالتالي سيتم تنفيذ الأمر التالي للأمر if، أما إذا كان الباقي لا يساوي صفراً فإن ذلك يعنى أن الرقم سيكون فردياً وبالتالي ففي هذه الحالة سيذهب التنفيذ إلى ما بعد الأمر else. إذن في هذه الصورة الجديدة من الأمر if نلاحظ أنه إذا كان الشرط محققا سيذهب التنفيذ إلى ما بعد الأمر if وينفذ الأمر أو الأوامر التالية له، أما إذا كان الشرط غير محقق فإن التنفيذ سيذهب إلى الأمر أو الأوامر بعد الأمر else وينفذها. لاحظ هنا أن الصورة if...else هي أمر واحد؛ ولذلك يجب ألا ننسى أن له نهاية واحدة وهي end.

الصورة الأخيرة للأمر if هي الصورة if...elseif والتي سنعيد كتابة برنامج التقديرات السابق باستخدامها كما يلي

```
%If statement
course_mark=input('Mark = ');
if (course_mark < 60 )
   course_grade='fail';
elseif (course_mark >=60 && course_mark < 70)
   course_grade='pass';
elseif (course_mark >=70 && course_mark < 80)
   course_grade='Good';
elseif (course_mark >=80 && course_mark < 90)
   course_grade='Very Good';
elseif (course_mark >=90 )
   course_grade='Excellent';
end
disp(['course_grade = '   course_grade]);
```

كاختبار لهذا البرنامج ننفذه في الحالتين التاليتين:

```
Mark = 70
course_grade = Good

Mark = 30
course_grade = fail
```

من الواضح أن هذه الصورة تكون مفضلة في حالة الاختيارات العديدة. لاحظ أيضا أنها كلها تعتبر بلوكاً واحداً للأمر if وتنتهي بنهاية end واحدة؛ لذلك فالذي يحدث عند تنفيذ هذا الشكل من الأمر if أنه يتم اختبار الشرط الأول التالي لـ if فإذا كان محققا يتم تنفيذ الأمر التالي له ثم الانتقال لما بعد الـ end أما إذا لم يكن الشرط الأول حقيقيا، فإن التنفيذ ينتقل لاختبار الشرط الثاني وبناء عليه إما أن ينفذ الأمر التالي له وينتقل لما بعد end وإما أن ينتقل إلى الشرط الثالث، وهكذا حتى ينتهي من جميع الشروط. لاحظ أن elseif تكتب كلمة واحدة. من المستحب أيضا في مثل هذه الأوامر أن يتم إدخال الأوامر إلى اليمين قليلا عن كلمة elseif حتى يبدو شكل هذه التركيبة واضحا من حيث البداية والنهاية وتكون سهلة المتابعة. إن هذه الصورة التي نستخدم فيها الأمر elseif تكون أفضل من الطريقة الأولى التي نستخدم العديد من الأوامر if...end. في هذه الصورة يتم اختبار جميع شروط الأوامر if...end حتى نصل إلى النهاية، بينما في الشكل الذي يستخدم elseif فإنه يتم اختبار الشروط حتى يتحقق أحدها حيث عندها يخرج من الأمر ولا يختبر باقي الشروط، لذلك فإن هذه الصورة elseif تكون أسرع في التنفيذ عن الصورة الأولى.

أوامر if المتداخلة Nested if

يمكن لأوامر if أن تتداخل أو يتم تعشيقها داخل بعضها لأي درجة كما في البرنامج التالي الذي يقرأ أرقاما يدخلها المستخدم، فإذا كان الرقم فردياً يخرج من البرنامج ويعرض كلمة فردى "odd"، وإذا كان الرقم زوجياً يقرر إذا كان هذا الرقم أقل من عشرة يعرض "even less than 10" وإذا لم يكن أقل من عشرة يعرض " even larger than 10".

```
%Nested if
x=input('Write a number    ');
if rem(x,2)==0
    if x<10
```

```
        disp('Even less than 10');
    else
        disp('Even larger than 10');
    end
else
    disp('Odd');
end
```

تنفيذ هذا البرنامج تحت الشروط المختلفة كما يلي :

Write a number 3
Odd

Write a number 100
Even larger than 10

Write a number 8
Even less than 10

يتكون هذا البرنامج من أمرين من النوع if...else...end الأمر الأول يحتضن بداخله أمر آخر من نفس النوع. لاحظ طريقة الكتابة وإدخال بعض الأوامر إلى الداخل بحيث يظهر كل else تتبع أي if وكل end تتبع أي if كذلك. في هذا البرنامج ؛ بعد إدخال الرقم يتم اختباره عن طريق الـ if الأولى، فإذا كان الرقم المدخل فردياً يذهب إلى آخر else وينفذ أوامرها ويخرج. أما إذا كان الرقم زوجياً فإنه يذهب إلى الـ if الداخلية ليختبر إذا كان الرقم أقل من ١٠ أم لا ويكتب الرسالة المناسبة له ويخرج.

الأمر switch...case...otherwise

هذه هي آخر صورة من أوامر الشرط والصورة العامة لهذا الأمر هي :

```
switch expression (scalar or string)
   case value1
      statements        % Executes if expression is
value1
   case value2
      statements        % Executes if expression is
value2
     .
     .
     .
   otherwise
      statements        % Executes if expression does not
match any case
end
```

حيث كل من كلمة switch و case و otherwise و end هي كلمات مفتاحية توضع كما هي وبنفس الصورة. التعبير expression التالي لكلمة switch يتم تنفيذه بحيث يؤول إلى قيمة واحدة. إذا كانت هذه القيمة تساوي القيمة value1 التالية لأول case فإن الأمر أو الأوامر التالية له يتم تنفيذها، وإذا كانت قيمة التعبير تساوي القيمة value2 التالية لثاني case فإن الأمر أو الأوامر التالية لهذه الـ case يتم تنفيذها، وهكذا بحيث إنه تبعا لقيمة التعبير expression سيتم تنفيذ case واحدة فقط. أما إذا كانت قيمة التعبير لا تساوي أي واحدة من القيم السابقة، فإن التنفيذ سيذهب إلى الأوامر التالية بعد كلمة otherwise وينفذها وبالتالي ينتهي الأمر.

البرنامج التالي يسألك عن رقم شهر معين، وهو يرد عليك باسم هذا الشهر:

```
%switch statement
month_number=input('write month number from 1 to 12
');
switch month_number
    case 1
            disp('this month is January');
    case 2
            disp('this month is February');
    case 3
            disp('this month is March');
    case 4
            disp('this month is April');
    case 5
            disp('this month is May');
    case 6
            disp('this month is June');
    case 7
            disp('this month is July');
    case 8
            disp('this month is August');
    case 9
            disp('this month is September');
    case 10
            disp('this month is October');
    case 11
            disp('this month is November');
    case 12
            disp('this month is December');
    otherwise
            disp('Impossible month number');
end
```

بعض الحالات لتنفيذ البرنامج ستكون كالتالي :

write month number from 1 to 12   3
this month is March

write month number from 1 to 12   9
this month is September

write month number from 1 to 12   0
Impossible month number

يمكن تخصيص أكثر من قيمة لكل case كما في المثال التالي الذي يأخذ رقم الشهر ويرد البرنامج في أي فصل من فصول السنة يوجد هذا الشهر :

```
%switch statement
month_number=input('write month number from 1 to 12
');
switch month_number
    case {12,1,2}
            disp('This season is winter');
    case {3,4,5}
            disp('this season is spring');
    case {6,7,8}
            disp('this season is summer');
    case {9,10,11}
            disp('this season is fall');
    otherwise
            disp('Impossible month number');
end
```

وتنفيذ هذا البرنامج لبعض الحالات سيكون كالتالي :

write month number from 1 to 12   4
this season is spring

write month number from 1 to 12   9
this season is fall

write month number from 1 to 12   2
This season is winter

write month number from 1 to 12   0
Impossible month number

جدول (٣,٣) يبين العمليات العلاقية أو النسبية التي تستخدم لتكوين الشروط المستخدمة في هذا النوع من الأوامر.

جدول (٣,٣). العمليات النسبية.

رمز العملية	العملية
<	أقل من
<=	أقل من أو يساوى
==	يساوى
~=	لا يساوى
>	أكبر من
>=	أكبر من أو يساوى

نؤكد هنا على الفرق بين علامة التساوي التي تتكون من علامتين تساوي == وعلامة التخصيص التي تتكون من علامة تساوي واحدة = . التعبير a=5 معناه ضع القيمة ٥ في المتغير a ولذلك نطلق عليها علامة تخصيص، لأنها تخصص أو تضع قيمة في متغير أو حتى متغير في متغير آخر مثل a=b والذي يعنى ضع قيمة المتغير b في المتغير a. أما الأمر if a==b مثلا فإنه يسأل عما إذا كانت a تساوي b أم لا ؛ ولكنه لا يغير قيمة المتغير a كما في أمر التخصيص.

(٣,٩) الحلقة

لكي نفهم هذا النوع من الحلقات سنقدم البرنامج التالي والذي يمثل لعبة التوقع. في أول أمر تم استخدام الدالة rand التي تعطي رقماً عشوائياً من صفر حتى واحد وتم ضربه في ١٠ وجمع واحد عليه ثم تم إهمال أي كسر فيه عن طريق الدالة floor ليكون الرقم العشوائي من واحد حتى ١٠ كرقم صحيح. بعد ذلك تم استخدام الأمر input لكي تدخل من خلاله توقعاتك لهذا الرقم حيث يتم وضعها في المتغير

guess. الأمر load splat يحمل ملفاً يحتوي صوتاً موسيقياً ستسمعه بعد أن تدخل توقعاتك عن طريق الأمر sound. هنا تبدأ الحلقة while والتي لها شرط وهو هل توقعاتك guess لا تساوي الرقم العشوائي matnum حيث طالما أنها لا تساوى، أي أن الشرط محقق، فإن التنفيذ سيدخل الحلقة ويسمعك الصوت بالأمر sound ثم ينظر، هل توقعاتك أعلى من الرقم العشوائي أم أقل عن طريق الأمر if...else؟ وفي كل حالة سيعطيك رسالة تفيد ذلك. بعد ذلك يسألك لتعطي توقعاً جديداً ويعود لبداية الحلقة while لينظر في هذا التوقع. تستمر هذه المحاولات إلى أن يصبح التوقع الذي أدخلته مساويا للرقم العشوائي، حيث عندها سيصبح شرط الحلقة غير محقق ونتيجة لذلك سيخرج التنفيذ من الحلقة إلى ما بعد الـ end الخاصة بها، حيث يحمل ملف صوت جديداً ويسمعك إياه بالدالة sound. اكتب هذا البرنامج في ملف M وحاول تنفيذه. ويمكنك طلب المساعدة عن الدوال load و sound لتعرف المزيد عنهم .

```
%While example
matnum = floor(10 * rand + 1);
guess = input( 'Your guess please: ' );
load splat
while guess ~= matnum
    sound(y, Fs)
if guess > matnum
    disp( 'Too high' )
else
    disp( 'Too low' )
end;
guess = input( 'Your next guess please: ' );
end
disp( 'At last!!!' );
load handel;
sound(y, Fs)
```

من ذلك نرى أن الصورة العامة للحلقة while يمكن كتابتها كالتالي :

```
while condition
        statements
end
```

حيث سيتم تنفيذ جسم الحلقة statements طالما أن الشرط condition يبقى حقيقياً. أي أنه يتم اختبار شرط الحلقة في بداية دخولها، ولذلك لابد أن تكون هناك قيم ابتدائية للمتغيرات التي تشكل شرط الحلقة condition قبل الدخول فيها كما حدث في برنامج التوقعات السابق حيث تم تنفيذ الأمر input قبل الدخول في الحلقة. أيضا في أثناء تنفيذ الحلقة لابد من تغيير متغيرات شرط الحلقة في كل مرة وإلا ستظل الحلقة تنفذ إلى مالانهاية، وفي برنامج التوقعات السابق كان الأمر input يتم تنفيذه داخل جسم الحلقة لتجديد متغير شرط الحلقة باستمرار.

الأمر break

يستخدم الأمر break للخروج من الحلقة for أو الحلقة while. يتم وضع هذا الأمر داخل الحلقات في الأماكن التي نخاف فيها من دخول الحلقة إلى تنفيذ لا نهائي؛ ولذلك من المستحسن أن نضع عداداً داخل الحلقة، وإذا وصل هذا العداد لقيمة معينة دون أن تنتهي الحلقة، نقوم بإنهاء الحلقة إجباريا باستخدام الأمر break، حيث بمجرد تنفيذه يخرج التنفيذ من حيز هذه الحلقة إلى خارجها.

الخطوات التالية توضح كيفية الدخول في حلقة لانهائية، حيث يطلب من المستخدم أن ينقر أي زر حتى يتم إدخال الحرف q وعندها يتم الخروج من البرنامج.

```
while(1)
    req = input('Press any Key to continue or "q" to
quit : ','s');
    if (req=='q')
        break;
    end
    disp(req);
end
```

(٣,١٠) الأرقام المركبة

عند التعامل مع الأرقام المعقدة أو المركبة هناك جزء حقيقي وجزء تخيلي. الماتلاب يعامل الأعداد المركبة بنفس الطريقة التي يعامل بها الأعداد العادية مع زيادة بعض الدوال الخاصة بتحديد الجزء الحقيقي والتخيلي .إذا فرضنا أن z هو عدد مركب و x و y هما عددان حقيقيان، و i هو عدد تخيلي، فمن الممكن تمثيل العدد المركب z كما يلي:

z=x+yi $\qquad i=\sqrt{-1}$

المتغير i والذي يساوي $\sqrt{-1}$ معرف مسبقا في برنامج الماتلاب على أنه المتغير المركب:

"the complex variable"

لاحظ عند كتابة :

```
>> i
ans =
      0 + 1.0000i
```

وهذا ينطبق على المتغيرj أيضا:

```
>> j
ans =
      0 + 1.0000i
```

كذلك يمكن الكتابة على هذا المتغير:

```
>> a=5+8i
a =
   5.0000 + 8.0000i
>> a=5+8*i
a =
   5.0000 + 8.0000i
>> i=4
i =
    4
>> a=5+8*i
a =
    37
```

لذلك يجب الحرص عند فرض أي متغير له الاسم i، ويجب التأكد من وضع قيمة ابتدائية للمتغير وإلا سنجد قيمته المستخدمة في البرنامج هي القيمة المعرفة من قبل الماتلاب.

فيما يلي بعض الدوال الخاصة بتعريف العدد المركب، إيجاد المرافق، تحديد الجزء الحقيقي و التخيلي، القيمة، الاتجاه:

complex, conj, real, imag, abs, angle, isreal

للتعرف على هذه الدوال؛ لاحظ مجموعة الأوامر التالية وناتج تنفيذ كل أمر من هذه الأوامر:

```
>> z=complex(5,9)
z =
  5.0000 + 9.0000i
>> conj(z)
ans =
  5.0000 - 9.0000i
>> real(z)
ans =
    5
>> imag(z)
ans =
    9
>> abs(z)
ans =
   10.2956
>> angle(z) %  phase angle in radians
ans =
   1.0637
>> angle(z)*180/pi  % to convert into degree
ans =
  60.9454
>> isreal(z)
ans =
    0
```

الأمثلة الآتية توضيح عمليات الجمع، الطرح، الضرب، القسمة في حالة الأعداد المركبة:

```
>> a=3+7j;
>> b=2+4i;
>> a+b
ans =
  5.0000 +11.0000i
>> a-b
ans =
  1.0000 + 3.0000i
>> a*b
ans =
 -22.0000 +26.0000i
>> a/b
ans =
  1.7000 + 0.1000i
```

فيما يلي مجموعة من التمارين المحلولة على ما تم تناوله بالشرح في هذا الفصل.

تمارين محلولة

٣-١ اكتب برنامجا يقرأ عدداً صحيحاً s ويحسب مجموع كل الأعداد الموجبة الأقل منه.

```
% Solution of exercise 3-1
s=input('Enter a positive integer number(s): ');
sum = 0;
for x=1:s
    sum = sum + x;
end
disp(['The summation of positive integers less than
',num2str(s),' = ',num2str(sum)])
```

٣-٢ اكتب دالة لحساب مكعب أي رقم m .

```
function [C]=num_cube(n)
C=n*n*n;
```

من مساحة لعمل ؛ نكتب الأمر التالي :

```
>> num_cube(2)
```

٣-٣ استخدم الدالة في التمرين السابق لحساب مجموع مكعبات الأرقام الموجبة الأقل من ٦.

```
% Solution of exercise 3-3
% Program to calculate sum of cubes of positive numbers
less than 6
sum = 0;
for m=1:6
    sum = sum + num_cube(m);
end
disp(['The summation = ',num2str(sum)])
```

٣-٤ اكتب برنامجا يقرأ عدداً صحيحاً n ويحسب المضروب (factorial)

$$n! = n*(n-1)*(n-2)*...*3*2*1$$

```
% Solution of exercise 3-4
n=input('Enter a positive integer number(n): ');
f=1;
for x=n:-1:1
    f = f * x;
end
disp(['The factorial of ',num2str(n),' = ',num2str(f)])
```

٣-٥ اكتب دالة لحساب مضروب أي عدد صحيح n.

```
% Solution of exercise 3-5
function f=myfact(n)
f=1;
if(n<=0)
    error('n must be a positive integer')
end
for x = n:-1:1
        f = f * x;
end
```

٣-٦ استخدم الدالة في التمرين السابق لكتابة برنامج لحساب وطباعة مضروب الأعداد من ١ إلى ١٠باستخدام:

١- جملة التكرار for

```
% Solution of exercise 3-6-1
for x=1:10
    f=myfact(x);
    disp(['The factorial of ',num2str(x),' =
',num2str(f)])
end
```

٢- جملة التكرار while

```
% Solution of exercise 3-6-2
x=1;
while x<=10
   f=myfact(x);
    disp(['The factorial of ',num2str(x),' =
',num2str(f)])
    x=x+1;
end
```

٣-٧ اكتب برنامجا لقراءة عددين حقيقيين (x,y) بينهما إشارة عملية حسابية (op) ثم يحسب ويطبع ناتج العددين تبعا للعملية الحسابية (+، -، *، /)

```
% Solution of exercise 3-7
s=input('Enter two numbers with an operation between
them','s')
x=str2num(s(1));
op=s(2);
y=str2num(s(3));
z=0;
if ( op == '+' )
   z=x+y;
elseif( op == '-' )
   z=x-y;
elseif( op == '*' )
    z=x*y;
elseif( op == '/' )
    z=x/y;
else
    disp('Undefined operation')
end
disp(['The result of
',num2str(x),op,num2str(y),'=',num2str(z)])
```

٣-٨ اكتب برنامجا يقرأ أطوال أضلاع مثلث ثم يطبع كلمة :

- Equilateral في حالة تساوي الأضلاع
- Isosceles في حالة متساوي الساقين
- Scalene في حالة اختلاف الأضلاع

```
% Solution of exercise 3-8
% Program to determine the triangle type
```

```
L1=input('Enter the first side length: ');
L2=input('Enter the second side length: ');
L3=input('Enter the thirs side length: ');
if (L1==L2)&&(L2==L3)
    disp('Equilateral')
elseif (L1==L2)||(L1==L3)||(L2==L3)
    disp('Isosceles')
else
    disp('Scalene')
end
```

لقد قدمنا في هذا الفصل عرضا سريعا لاستخدام ماتلاب كلغة برمجة عامة الأغراض، وإن المتمرس على البرمجة بلغة C سيشعر بعدم وجود فروق بين اللغتين. ولقد راعينا أن تكون المراجعة سريعة اعتمادا على أن المستخدم لديه فكرة عن لغة C وإن كان ذلك غير ضروري لأن ماتم عرضه في هذا الفصل يعتبر كافيا لوضع المستخدم على بداية طريق البرمجة بالماتلاب. المراجع [١-٨] تقدم شرحا مفصلا للبرمجة باستخدام الماتلاب مع العديد من الأمثلة والتطبيقات.

الفصل الرابع

المصفوفات في ماتلاب

(٤,١) مقدمة

المصفوفة هي مجموعة من البيانات الموضوعة في صورة ثنائية الأبعاد، أي في صورة صفوف وأعمدة أو على هيئة جدول. يعتبر ماتلاب كما رأينا مسبقا وكما يعكس اسمه بأنه هو لغة المصفوفات حيث إن MATLAB هي اختصار لعبارة MATrix LABoratory أو معمل المصفوفات حيث كما رأينا أن أي متغير في ماتلاب ينظر إليه على أنه مصفوفة. سنتعلم في هذا الفصل كيفية تعامل الماتلاب مع المصفوفات وكيفية بدأها وتعريفها، ثم ننتقل إلى العمليات المختلفة على المصفوفات من جمع وطرح وضرب وعكس وغيره، ثم بعد ذلك سنتعلم كيفية التعامل مع سلاسل الأحرف.

(٤,٢) إنشاء المصفوفات وبعض العمليات البسيطة

يمكن إنشاء المصفوفة في ماتلاب كما يلي :

```
>> a=[1 2 3;4 5 6]
a =
   1   2   3
   4   5   6
```

حيث a هو اسم المصفوفة والقوس المربع [هو فتح المصفوفة ثم نبدأ بإدخال عناصر المصفوفة عنصراً بعد الآخر ويفصل كل عنصر عن الثاني إما بمسافة أو فاصلة. بعد الانتهاء من إدخال الصف الأول ننهيه بفاصلة منقوطة، ثم نبدأ بإدخال عناصر الصف الثاني وننهيه بفاصلة منقوطة، وهكذا إلى أن ننهى جميع صفوف المصفوفة، حيث ننهيها بالقوس المربع المقفول] . إذا كنت في نافذة الأوامر command window فإنه بمجرد أن تضرب Enter ستظهر لك المصفوفة بشكلها الطبيعي.

```
>> x=[7 8 9]
x =
   7   8   9

>> a=[a;x]
a =
   1   2   3
   4   5   6
   7   8   9
```

هنا تم تعريف المتغير x كمصفوفة من صف واحد له ثلاثة عناصر ثم استخدم الأمر a=[a;x] لوضع المتغير x كصف في نهاية المصفوفة a لتصبح مصفوفة من ثلاثة صفوف وثلاثة أعمدة. بدلا من استخدام الفاصلة المنقوطة كفاصل بين الأعمدة يمكن استخدام الـ Enter لهذا الغرض كما يلي:

```
>> a=[1 2 3
4 5 6
7 8 9]
a =
   1   2   3
   4   5   6
   7   8   9
```

التعبير عن عناصر المصفوفة

العناصر المختلفة في أي مصفوفة يمكن التعبير عنها كالتالي: a(2,3) والذي يعنى العنصر الموجود في الصف الثاني والعمود الثالث من المصفوفة a. ولذلك لو ذهبنا الآن إلى حقل نافذة الأوامر وكتبنا a(2,3) ثم Enter سيرد علينا ماتلاب بقيمة هذا العنصر كما يلي :

```
>> a(2,3)
ans =
   6
```

وهكذا يمكن عرض قيمة أي عنصر من عناصر المصفوفة. لاحظ استخدام الأقواس المربعة [ ] للدلالة على بداية ونهاية المصفوفة بينما القوس المستدير ( ) للدلالة على عنصر مصفوفة. يمكنك تخصيص قيما عددية لعناصر المصفوفة المنفردة كما يلي :

```
>> a(2,2)=50;
>> a
a =
   1    2    3
   4   50    6
   7    8    9
```

حيث قد غيرنا قيمة العنصر الموجود في الصف الثاني والعمود الثاني من ٥ إلى ٥٠. ماذا لو طلبنا عرض عنصر خارج أبعاد المصفوفة مثلا a(2,4) ، انظر للتالي :

```
>> a(2,4)
??? Attempted to access a(2,4); index out of bounds because size(a)=[3,3].
```

حيث رد ماتلاب بأننا نحاول الوصول إلى عنصر غير موجود في المصفوفة ، وأن آخر عنصر هو a(3,3). نحذر هنا من شيء غاية في الخطورة وهو أننا على فرض بطريق الخطأ كتبنا أمر التخصيص التالي a(2,4)=12 ، انظر ماذا فعل الماتلاب :

```
>> a(2,4)=12
a =
   1    2    3    0
   4   50    6   12
   7    8    9    0
```

لقد مدد الماتلاب أبعاد المصفوفة بزيادة عمود لها لتستوعب هذا العنصر الجديد مع وضع أصفار في العناصر المضافة الأخرى، بحيث أصبحت أبعاد المصفوفة 3x4 بدلا من 3x3.

(٤,٣) دوران المصفوفة

دوران المصفوفة يقصد به جعل الصفوف أعمدة والأعمدة صفوفا. رمز هذه العملية هو علامة التنصيص الأحادية (') والمثال التالي يوضح ذلك:

```
>> a=[1:3;4:6]
a =
    1    2    3
    4    5    6
>> b=a'
b =
    1    4
    2    5
    3    6
```

حيث المصفوفة b نتجت من تدوير المصفوفة a بحيث أصبح الصف الأول في a هو العمود الأول في b والصف الثاني في a هو العمود الثاني في b. لاحظ طريقة إنشاء المصفوفة a باستخدام عملية النقطتين، حيث الصف الأول كتب ٣:١ ثم الفاصلة المنقوطة ثم ٦:٤ وهذا ممكن كما رأينا طالما أن الفرق بين عنصر والتالي له هو فرق ثابت.

إن عملية النقطتين تعتبر من العمليات التي ينفرد بها ماتلاب عن باقي لغات البرمجة والتي يمكن استخدامها بكثرة. انظر لما يلي:

```
>> a=[1:3;4:6;7:9]
a =
    1    2    3
    4    5    6
    7    8    9
>> b=a(2:3,1:2)
b =
    4    5
    7    8
```

حيث تم إنشاء المصفوفة a بالكامل باستخدام معامل النقطتين، ثم تم إنشاء المصفوفة b من الصفين ٢ و ٣ والعمودين ١ و ٢ من المصفوفة a كما رأينا، حيث تم إنشاء المصفوفة b كجزء من المصفوفة a.

```
>> b=a(2,:)
b =
   4   5   6
```

حيث تم تشكيل المصفوفة b من الصف الثاني وكل الأعمدة في المصفوفة a.

```
>> a(1:2,2:3)=ones(2)
a =
   1   1   1
   4   1   1
   7   8   9
```

حيث نجعل الصفوف واحد واثنين والأعمدة اثنين وثلاثة في المصفوفة a تساوي وحايد كما رأينا. المصفوفة ones(n) تعطي مصفوفة مربعة nxn كل عناصرها وحايد كما يلي :

```
>> b=ones(3)
b =
   1   1   1
   1   1   1
   1   1   1
```

كما أن المصفوفة zeros(n) تعطي مصفوفة مربعة كل عناصرها أصفار كما يلي :

```
>> zeros(3)
ans =
   0   0   0
   0   0   0
   0   0   0
```

من ذلك نرى أن معامل النقطتين يستخدم للتعبير عن رقم صف أو رقم عمود في عنصر من عناصر مصفوفة، وفي حالة وجودة وحدة فإن ذلك يعني كل أو كل الأعمدة. فمثلا a(1,:)، يقصد به العناصر المكونة من الصف الأول وكل الأعمدة، بينما a(:,2) فيقصد به كل الصفوف والعمود الثاني من المصفوفة a. هذه العملية تكون مفيدة جدا في إنشاء بعض الجداول كما في المثال التالي الذي ينشىء جدولا من جيب وجيب تمام الزوايا من صفر حتى ١٨٠ بفارق ٣٠ درجة.

```
%table of sin and cos using colon operator
x=[0:30:180]';
table(:,1)=x;
table(:,2)=sin(x*pi/180);
table(:,3)=cos(x*pi/180);
disp(table);
```

وأما الجدول الناتج من تنفيذ هذا البرنامج فهو كالتالي:

```
        0         0    1.0000
  30.0000    0.5000    0.8660
  60.0000    0.8660    0.5000
  90.0000    1.0000    0.0000
 120.0000    0.8660   -0.5000
 150.0000    0.5000   -0.8660
 180.0000    0.0000   -1.0000
```

```
>> b=zeros(3);
>> a=ones(3);
>> a(:,[1 3]) = b(:,[1 2])
a =
   0   1   0
   0   1   0
   0   1   0
```

حيث جعلنا كل الصفوف في العمودين واحد وثلاثة في المصفوفة a تساوي كل الصفوف في العمودين واحد واثنين في المصفوفة b.

يمكن استخدام عملية النقطتين في إجراء بعض العمليات الحسابية على صفوف وأعمدة أي مصفوفة كما يلي:

```
>> a=ones(3);
>> a(:,2)=a(:,2)+2*a(:,3)
a =
    1    3    1
    1    3    1
    1    3    1
```

حيث تم جمع حاصل ضرب كل عناصر العمود الثالث في ٢ مع ما يناظرها من عناصر العمود الثاني في نفس المصفوفة a.

الأمر sum(x)

يجمع كل عناصر المتجه x:

```
>> x=[1 2  3];
>> sum(x)
ans =
    6
```

إذا كانت x مصفوفة ثنائية الأبعاد، فإن sum(x) يجمع محتويات كل عمود على حده، ويعطي في الخرج متجهاً من هذه المجاميع كما يلي:

```
>> x=ones(3);
>> sum(x)
ans =
    3    3    3
```

يمكن استخدام معامل النقطتين لحساب مجموع عدد معين من عناصر أي متجه كما يلي :

```
>> a=[1 2 3 4 5];
>> x=sum(a(3:end))
x =
   12
```

حيث end في الأمر sum تعني نهاية المتجه a ، وعلى ذلك فالأمر السابق يحسب مجموع عناصر المتجه a بدءا من العنصر الثالث حتى النهاية.

هناك أمر آخر في الماتلاب يعطي المجموع التراكمي ، وهو الأمر cumsum والذي نوضحه كما يلي :

```
>> a=[1 2 3 4 5];
>> c=cumsum(a)
c =
   1    3    6   10   15
```

في هذه الحالة يكون الناتج ليس هو مجموع المتجه a كما في الأمر sum ولكن كل عنصر في المتجه الناتج c يقابل مجموع العنصر المقابل له في المتجه a وجميع العناصر السابقة له.

```
>> a=[1 2 3;4 5 6;7 8 9]
a=
   1    2    3
   4    5    6
   7    8    9
>> c=cumsum(a)
c =
   1    2    3
   5    7    9
  12   15   18
```

يمكن استخدام معامل النقطتين في إلغاء بعض الصفوف أو بعض الأعمدة كما يلي :

```
>> a=[1 2 3;4 5 6;7 8 9]
a =
   1   2   3
   4   5   6
   7   8   9
>> a(:,2)=[]
a =
   1   3
   4   6
   7   9
```

حيث بعد إنشاء المصفوفة a تم إلغاء العمود الثاني فيها. وفيما يلي سنلغى الصف الأول في المصفوفة a الناتجة.

```
>> a(1,:)=[]
a =
   4   6
   7   9
```

لا يمكن إلغاء عنصر معين من مصفوفة، لذلك فالأمر a(2,2)=[] خطأ، ولن يقبله ماتلاب وسيعطي عليه رسالة خطأ.

(٤.٤) بعض دوال المصفوفات الأولية

هناك بعض المصفوفات الأولية التي تستخدم في الكثير من التطبيقات، والتي رأينا بعضها مثل المصفوفة zeros(n) والتي تعطي مصفوفة مربعة n×n كل عناصرها أصفار، والمصفوفة ones(n) التي تعطي مصفوفة n×n أيضا كل عناصرها وحايد. هناك أيضا المصفوفة eye(n) التي تعطي مصفوفة قطرية مربعة قطرها الرئيس وحايد وباقي عناصرها أصفار كما يلي :

```
>> eye(3)
ans =
   1   0   0
   0   1   0
   0   0   1
```

الأمر help elmat بمجرد كتابته في نافذة الأوامر command window سيعرض العديد من هذه المصفوفات والدوال والتي لا مجال لعرضها كلها هنا ولكن نترك القارئ للمحاولة معها وتجريبها بنفسه. من هذه المصفوفات التي قد نستخدمها المصفوفة rand(n) التي تعطي مصفوفة مربعة n×n قيم عناصرها عشوائية من صفر حتى واحد كما يلي :

```
>> rand(3)
ans =
   0.5469    0.1576    0.4854
   0.9575    0.9706    0.8003
   0.9649    0.9572    0.1419
```

المصفوفة rand(m,n) تعطي مصفوفة عشوائية مكونة من m من الصفوف و n من الأعمدة. بينما المصفوفة rand بدون أي معاملات فتعطي مصفوفة من عنصر واحد (قيمة واحدة) بقيمة عشوائية من صفر حتى واحد.

هناك المصفوفة randn(n) مثل نظيرتها rand(n) ولكن القيمة العشوائية تكون بقيمة متوسطة صفراً، أي أنها تعطي قيمة عشوائية سالبة وموجبة بحيود معياري مقداره واحد.

```
>> randn(3)
ans =
  -0.4326    0.2877    1.1892
  -1.6656   -1.1465   -0.0376
   0.1253    1.1909    0.3273
```

هناك بعض المصفوفات الخاصة مثل المصفوفة magic(n) التي تعطي مصفوفة تتغير قيمها من صفر حتى n^2 ومجموع أي صف فيها أو أي عمود أو أي قطر كلها تكون متساوية وتعطي نفس المجموع كما في المصفوفة التالية :

```
>> magic(3)
ans =
    8    1    6
    3    5    7
    4    9    2
```

التي هي مصفوفة مربعة ٣X٣ مجموع أي صف أو أي عمود أو أي قطر فيها يساوي ١٥.

(٤,٥) العمليات الحسابية على المصفوفات

عندما تكون مصفوفة معاملاً لأي دالة حسابية أو مثلثية فإن هذه الدالة يتم إجراؤها على جميع عناصر المصفوفة منفردة. فمثلا يمكن ضرب أو قسمة كل عناصر المصفوفة في ثابت كما يلي :

```
>> a=ones(3)
a =
   1   1   1
   1   1   1
   1   1   1
>> a=2*a
a =
   2   2   2
   2   2   2
   2   2   2
```

هنا سنجري الدالة sin على جميع عناصر مصفوفة كما يلي :

```
>> a=[0:30:120]
a =
   0   30   60   90   120
>> b=sin(a*pi/180)
b =
      0   0.5000   0.8660   1.0000   0.8660
```

وكذلك عملية الأس على كل عنصر من عناصر المصفوفة على حدة باستخدام عمليات النقط التي تم شرحها مسبقا مثل .* و .^ و ./ وغيرها :

```
>> a=[1 2 3;4 5 6]
a =
   1   2   3
   4   5   6
>> b=a.^2
b =
   1   4   9
  16  25  36
```

نذكر هنا على أنه هناك بعض الدوال التي عندما يتم إجراؤها على مصفوفة فإنها تُجرى على أعمدة المصفوفة فقط. مثال ذلك الدالة sum التي تعطي مجموع أعمدة أي مصفوفة إذا كانت تُجرى على مصفوفة ثنائية الأبعاد.

إذا كنت غير متأكد من أن الدالة التي تستخدمها تعمل على العناصر أو على الأعمدة فعليك طلب المساعدة help من ماتلاب وهو سيخبرك.

هناك العمليات الحسابية على زوج من المصفوفات أو أكثر مثل جمع وطرح أكثر من مصفوفة. في هذه العمليات لابد أن تكون المصفوفات متساوية الأبعاد. يمكن جمع وطرح مصفوفتين كما يلي :

```
>> a=ones(3);
>> b=ones(3);
>> c=a+b
c =
    2    2    2
    2    2    2
    2    2    2
>> d=a-b
d =
    0    0    0
    0    0    0
    0    0    0
```

هناك نوع من ضرب مصفوفتين، وهو ضرب كل عنصر في مصفوفة في نظيره في المصفوفة الأخرى كما يلي :

```
>> a=[1 2 3;4 5 6]
a =
    1    2    3
    4    5    6
>> b=[6 5 4;3 2 1]
b =
    6    5    4
    3    2    1
>> c=a.*b
c =
    6   10   12
   12   10    6
```

الضرب الحسابي لمصفوفتين يعتبر من أهم العمليات الحسابية التي تُجرى على مصفوفتين والتي تستخدم في تحليل الدوائر الكهربية وحل المعادلات الخطية والتحويل بين نظم الإحداثيات المختلفة. هذا النوع من الضرب يكتب على الصورة c=a*b أو c=ab حيث العنصر c_{ij} يساوي حاصل ضرب الصف i في المصفوفة a في العمود j في المصفوفة b. ولذلك فإنه من شروط هذا النوع من الضرب أن يكون عدد صفوف المصفوفة a يساوي عدد أعمدة المصفوفة b. يجب أن نتذكر أنه بناء على ذلك فإن ab لا تساوي ba، أي أن عملية الضرب هذه ليست تبادلية. كمثال لهذا النوع من الضرب انظر للمثال التالي:

$$\begin{bmatrix}1 & 2\\3 & 4\end{bmatrix}\times\begin{bmatrix}5 & 6\\0 & -1\end{bmatrix}=\begin{bmatrix}5 & 4\\15 & 14\end{bmatrix}$$

لترى هنا كيف يستخدم الماتلاب في تنفيذ هذا النوع من الضرب، حيث سنضرب المصفوفة a المكونة من صفين وثلاث أعمدة في المصفوفة b المكونة من ثلاث صفوف وعمودين لتعطي المصفوفة c المكونة من صفين وعمودين كما يلي:

```
>> a=[1 2 3;4 5 6]
a =
   1   2   3
   4   5   6
>> b=[1 2;3 4;5 6]
b =
   1   2
   3   4
   5   6
>> c=a*b
c =
  22  28
  49  64
```

إذن، هناك فرق كبير بين حاصل الضرب a.*b وحاصل الضرب a*b .

عملية تربيع مصفوفة تعنى ضرب المصفوفة في نفسها ؛ ولذلك لابد أن تكون المصفوفة مربعة ولذلك فإن العملية ^ تكافئ العملية * ، كمثال على ذلك :

```
>> a=[1 2;3 4]
a =
    1    2
    3    4
>> b=a^2
b =
    7   10
   15   22
```

أيضا نذكر هنا على الفرق بين العملية a^2 والعملية a.^2 والتي تقوم بتربيع كل عنصر من عناصر المصفوفة.

الأمر inv(x) المصفوفة العكسية (matrix inversion) للمصفوفة المربعة x كما في المثال التالي :

```
>> a = [1 2; 3 4]
a =
    1    2
    3    4
>> y = inv(a)
y =
  -2.0000    1.0000
   1.5000   -0.5000
```

الأمر diag(x) يعطي العناصر الموجودة في القطر الرئيس للمصفوفة x. كمثال على ذلك :

```
>> a=[1 2 3;4 5 6;7 8 9]
a =
    1    2    3
    4    5    6
    7    8    9
>> diag(a)
ans =
    1
    5
    9
```

والأمر fliplr(x) الذي يعكس المصفوفة x من الشمال لليمين، بإجراء هذا الأمر على نفس المصفوفة a السابقة نحصل على التالي:

```
>> fliplr(a)
ans =
   3   2   1
   6   5   4
   9   8   7
```

والأمر flipud(x) يعكس المصفوفة x من فوق لتحت، وبإجراء هذه العملية على نفس المصفوفة a السابقة نحصل على التالي:

```
>> flipud(a)
ans =
   7   8   9
   4   5   6
   1   2   3
```

ثم الأمر rot90(x) الذي يدور المصفوفة حول نفسها بمقدار ٩٠ درجة، وبإجراء هذه العملية على نفس المصفوفة a نحصل على التالي:

```
>> rot90(a)
ans =
   3   6   9
   2   5   8
   1   4   7
```

ثم الأمر tril(x) الذي يعطي المثلث الأسفل من المصفوفة x، وبإجراء هذه العملية على نفس المصفوفة a نحصل على التالي:

```
>> tril(a)
ans =
   1   0   0
   4   5   0
   7   8   9
```

وأخيرا، الأمر triu(x) الذي يعطي المثلث الأعلى من المصفوفة x، وبإجراء هذه العملية على نفس المصفوفة a نحصل على التالي:

```
>> triu(a)
ans =
   1   2   3
   0   5   6
   0   0   9
```

يمكن حساب قيمة المحددة المصاحبة لأي مصفوفة من خلال الأمر det(x) الذي يرد عليك بقيمة هذه المحددة. يحتوي الماتلاب على العديد من الدوال التي تستخدم في الجبر الخطى وجبر المصفوفات والتي يصعب ذكرها بالتفصيل هنا لما تحتاجه من مساحة ووقت.

```
>> a=[1 2;3 4]
a =
    1    2
    3    4

>> det(a)
ans =
   -2
```

(٤,٦) سلاسل الأحرف

سلسلة الأحرف هي مجموعة من الأحرف أو حتى حرف واحد، والتي قد تمثل اسم شخص مثلا أو تعليق. يتعامل ماتلاب مع سلسلة الأحرف على أنها متجه أو مصفوفة أحادية البعد وكل عنصر فيها يمثل حرفا من أحرف هذه السلسلة. مثلا:

```
>> s='mohamed'
s =
mohamed
```

حيث تم إدخال السلسلة mohamed كقيمة للمتغير s ولاحظ أنه لابد من وضع هذه السلسلة بين علامتي التنصيص الأحادية لإخبار ماتلاب أن هذا المتغير ليس متغيرا رقميا، ولكنه متغير حرفي يحتوي سلسلة أحرف. تعال نستعلم من ماتلاب كيفية تعامله وتخزينه لهذه السلسلة، وذلك باستخدام الأمر whos كما يلي:

```
>> whos s
  Name      Size          Bytes  Class     Attributes
  s         1x7              14  char
```

حيث تبين أن المتغير s هو متغير حرفي char يتكون من ٧ أحرف في صورة مصفوفة أحادية من سبعة عناصر ويشغل ١٤ بايت. انظر للأمر التالي:

```
>> s(7:-1:1)
ans =
demahom
```

حيث تم عرض المصفوفة s بالعكس بدءا من آخر عنصر حتى أول عنصر.

كما ذكرنا، فإن السلسلة كمتغير لابد أن توضع بين علامتي تنصيص أحادية، ولكن ماذا لو أن السلسلة نفسها تحتوي إحدى علامات التنصيص، في هذه الحالة يتم تكرار علامة التنصيص كما يلي:

```
>> s='8 O''clock'
s =
8 O'clock
```

يمكن استخدام الأمر input لإدخال سلاسل الأحرف كما يلي:

```
>> name=input('Enter your name   ')
Enter your name   Mohamed
??? Error using ==> input
Undefined function or variable 'Mohamed'.

Enter your name   'Mohamed'
name =
Mohamed
```

لاحظ أنه عندما سألك ماتلاب لإدخال اسمك؛ وتم إدخال الاسم بدون علامات تنصيص رفض ماتلاب وأعطى رسالة خطأ، ولكن بإدخال الاسم بين علامتي التنصيص تم قبوله وعرضه كما رأينا. من السهل جدا أن ينسى الشخص أن يكتب علامات التنصيص خاصة إذا كان مستخدم البرنامج غير متمرس على الماتلاب. يمكن تجنب شرط كتابة علامات التنصيص عند إدخال سلسلة الأحرف، وذلك بزيادة 's' في الأمر input كما يلي:

```
>> name=input('Enter your name   ','s')
Enter your name   Mohamed
name =
Mohamed
```

حيث هذه المرة تم قبول الاسم بدون علامات التنصيص كما رأينا نتيجة وضع 's' ضمن الأمر input.

يمكن مجاورة أو لصق concatenation سلسلتي أحرف، وذلك باستخدام أوامر المصفوفات كما يلي:

```
>> first_name='Mohamed ';
>> last_name='Eladawy';
>> full_name=[first_name  last_name]
full_name =
Mohamed  Eladawy
```

في هذا المثال تم عمل متجه اسمه full_name مكون من المتجهين first_name و last_name.

يتم تسجيل أي حرف في ماتلاب في ١٦ بتاً، والأكواد من ١ حتى ١٢٧ تمثل الأحرف الهجائية وهذه هي الشفرة ASCII المعروفة. فمثلا الأكواد الستعشرية من ٦٥ حتى ٩٠ تمثل الأحرف الكبيرة من A حتى Z، بينما الأكواد من ٩٧حتى ١٢٢ تمثل الأحرف الصغيرة a حتى z ، وهذه الأكواد كلها موجودة في العديد من المراجع وفي أماكن عديدة على الإنترنت. يمكنك أن تتعرف شفرة الأسكى لأي سلسلة باستخدام الدالة double كما يلي:

```
>> double('mohamed')
ans =
  109  111  104  97  109  101  100
```

حيث ١٠٩ تمثل شفرة الحرف m و ١١١ تمثل شفرة الحرف o وهكذا. الدالة char() تقوم بعكس ما تقوم به الدالة double حيث تعطي الحرف الهجائي المقابل لأي شفرة من شفرات الأسكى كما يلي:

```
>> char(80:85)
ans =
PQRSTU
```

سلسلة الأحرف مثلها مثل أي متجه يمكن أن تستخدم في الكثير من العمليات الحسابية كما يلي :

```
>> s='a';
>> double('a')
ans =
   97
>> s=s+1
s =
   98
>> char(98)
ans =
b
```

حيث تم وضع الحرف a في المتغير s، ثم عرضنا شفرته الأسكى وهي ٩٧، ثم تم زيادة واحد على المتغير s وبالتالي أصبحت الشفرة الأسكى الموجودة في s هي ٩٨ ثم عرضنا الحرف المقابل لهذه الشفرة وهو الحرف b.

يمكن مقارنة سلسلتي أحرف بطريقتين، الأولى باستخدام العمليات العلاقية < و > وغيرها كما يلي :

```
>> s1='ant';
>> s2='bny';
>> b=s1<s2
b =
   1   0   1
>> c=s2<s1
c =
   0   0   0
```

حيث المتغير b=s1<s2 يقارن حرف بحرف من اليسار ويعطي إجابة منطقية واحداً أو صفراً لكل عملية مقارنة. أولا ، هل الحرف a في السلسلة الأولى أقل من الحرف b في السلسلة الثانية؟ بالطبع ، الإجابة نعم ؛ وبالتالي فإن ماتلاب يضع ١ في المتغير b. ثم ،

هل الحرف الثاني n في المتجه الأول أقل من الحرف الثاني n في المتجه الثاني؟ بالطبع، الإجابة لا، وبالتالي؛ فإن ماتلاب سيضع صفراً في المتغير b. وهكذا تستمر المقارنة حتى آخر حرف. انظر إلى المتغير c السابق وحدد لماذا يحتوي على ثلاثة أصفار. لاحظ أنه في كل هذه المقارنات تتم المقارنة على شفرات الأسكي.

هناك الدالة strcmp التي تقارن سلسلتين وتعطي ١ إذا كانت السلسلتان متطابقتين تماما، وتعطي صفرا إذا كانت السلسلتان مختلفتين كما يلي:

```
>> strcmp(s1,s2)
ans =
    0
>> s2='ant'
s2 =
ant
>> strcmp(s1,s2)
ans =
    1
```

حيث إن أمر المقارنة الأول قارن السلسلة الأولى 's1='ant والسلسلة الثانية 's2='bny فكانت النتيجة صفرا لأن السلسلتين غير متساويتين، أما عندما جعلنا السلسلة الثانية تساوي الأولى ونفذنا أمر المقارنة كانت النتيجة ١ لتساوى السلسلتين.

الدالة blanks تولد سلسلة أحرف فارغة كما يلي:

```
>> c=blanks(5)
c =؟
>> size(c)
ans =
    1    5
```

حيث تم إنشاء السلسلة الفارغة من خمسة أحرف، ولكي نتأكد منها عرضنا طول هذه السلسلة باستخدام الأمر size الذي بين أن c متجه أحادي من خمسة عناصر.

الدالة deblank تحذف العناصر الفارغة من أي سلسلة أحرف كما يلي :

```
>> s='what is your name       '
s =
what is your name
>> size(s)
ans =
    1    24
>> b=deblank(s)
b =
what is your name
>> size(b)
ans =
    1    17
```

حيث هنا تم إنشاء السلسلة s التي تساوي العبارة 'what is your name   ' المنتهية بعدد من الأحرف الفارغة والتي كان طولها ٢٤ حرفا. بعد ذلك أنشأنا المصفوفة b التي تساوي المصفوفة s بعد إزالة الأحرف الفارغة في نهايتها فأصبح طولها ١٧ حرفا كما رأينا.

الدالة lower والدالة upper تحولان أي سلسلة من أحرف صغيرة إلى أحرف كبيرة أو العكس كما يلي :

```
>> x='ABCDEF';
>> lower(x)
ans =
abcdef
```

الدالة eval

يمكن وضع أي تعبير أو دالة يمكن تنفيذها كنص أو كسلسلة أحرف في أحد المتغيرات النصية بحيث يمكن النداء على الدالة eval التي يكون معاملها هو المتغير الحرفي فيقوم ماتلاب بتنفيذ هذه الدالة أو التعبير الموجود داخل هذا المتغير النصي. انظر إلى المثال التالي :

```
>> x='a*b';
>> a=2;
>> b=3;
>> c=eval(x)
c =
   6
```

حيث المتغير النصي x يحتوي عملية ضرب المتغير a في المتغير b كنص موضوع بين علامتي تنصيص. بعد ذلك حددنا a=2 و b=3 ثم الأمر c=eval(x) حيث سيقوم ماتلاب بقراءة النص الموجود في المتغير النصي x على أنه عملية حسابية يقوم بتنفيذها تبعا لقيمة كل من a و b ويعرض النتيجة في المتغير c كما سبق.

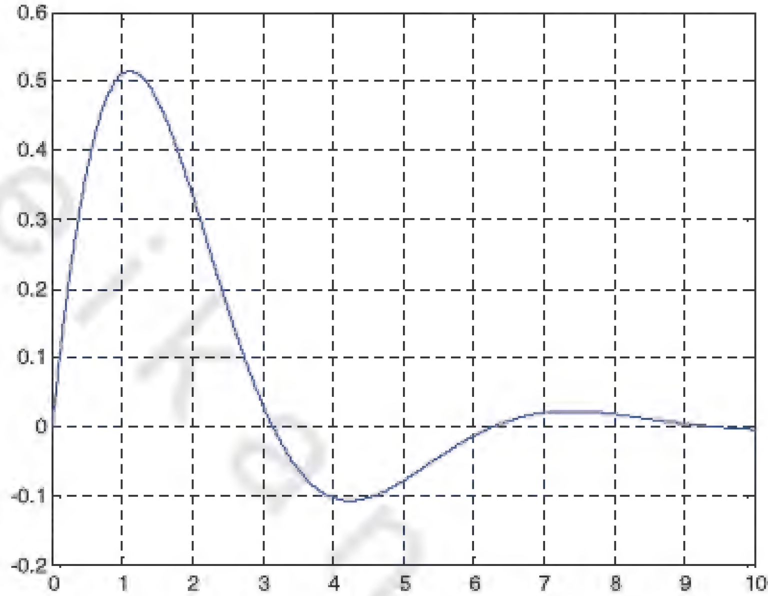
انظر لهذا المثال أيضا كتطبيق على الدالة eval :

```
% The eval function
f = input( 'Enter function (of x) to be plotted: ',
's');
x = 0:0.01:10;
plot(x, eval(f)),grid
```

الأمر input يطلب منك إدخال الدالة التي تريد رسمها كنص، وكما ذكرنا فقد وضعنا الحرف 's' لكي تعفى من وضع ما تريد كتابته بين علامتي تنصيص. لقد نفذنا البرنامج السابق مع الدالة المراد رسمها لقيم x المحددة في البرنامج كما يلي :

```
Enter function (of x) to be plotted:   exp(-0.5*x) .* sin(x)
```

وكان شكل الدالة كما في شكل (٤.١).



شكل (٤.١) تطبيق لرسم دالة باستخدام الأمر eval.

الأمر eval يعتبر من الأوامر المهمة عند كتابة البرامج، حيث يمكن أن يكون المتغير النصي متغيرا تبعا لمتطلبات البرنامج وليست ثابتاً من بدايته. في معظم الأحيان عند استخدام الأمر eval يكون المتغير النصي ناتجا عن تجميع بعض سلاسل الأحرف والمتغيرات - التي تتغير قيمتها مع تنفيذ البرنامج ووفقا لمدخلاته - تكون موجودة داخل قوسين مربعين. المثال التالي يوضح كيفية الحصول على مجموعة من المصفوفات الخاصة magic التي تحدثنا عنها سابقا.

```
for n = 1:12
    eval(['M' num2str(n) ' = magic(n)'])
end
```

لاحظ أن المتغير النصي الموجود بين القوسين المربعين ما هو إلا تجميع لثلاث سلاسل أحرف. الأولى اسم المتغير M ، والثانية هي تحويل الرقم n من صورته الرقمية إلى شكله الحرفي. أما السلسلة الأخيرة =magic(n) فهي عبارة عن اسم المصفوفة المطلوب الحصول عليها. لاحظ وجود علامة = وفكر، لماذا هي موجودة؟

في حالة n=1 فإن المتغير النصي عبارة عن magic(n)=M1 أما عند n=2 فإن المتغير النصي يكون magic(n)=M2 وهكذا. هذا موضح كناتج لتنفيذ الأمر التالي :

```
<<n=6
<<['M' num2str(n) ' = magic(n)']
ans=
M6 = magic(n )
```

تمارين محلولة

٤-١ اكتب برنامجا لقراءة ثلاثة أعداد من النوع الصحيح، ثم رتب هذه الأعداد تصاعديا باستخدام الدالة.

```
% Solution of exercise 4-1
% Program to order 3 integer numbers
b(1:3)=0;
for x=1:3
    a=input('Please enter an integer: ');
    b(1,x)=a;
end
disp(['You entered the numbers :', num2str(b(1)),' , ',
num2str(b(2)),' and ', num2str(b(3))])
b_sorted=sort(b)
```

٤-٢ اكتب برنامجا لقراءة أربعة متغيرات حقيقية، ثم استخدم الدالة لإيجاد أكبر قيمة من هذه المتغيرات.

```
% Solution of exercise 4-2
% Program to find the maximum of 4 real numbers
z(1:4)=0;
for x=1:4
    a=input('Enter a real number : ');
    z(1,x)=a;
end
disp(['You entered the real numbers :', num2str(z(1)),'
, ', num2str(z(1)),' , ', num2str(z(3)),' and ',
num2str(z(4))])
z_max=max(z)
```

٤-٣ اكتب برنامجا يقرأ مصفوفة أحادية البعد مكونة من ٢٠ رقماً ثم يقوم البرنامج بطباعة الأرقام التي هي أكبر من ١٠ مع طباعة أماكن تواجدها ضمن المصفوفة.

```
% Solution of exercise 4-3
m(1:20)=0;
for i=1:20
    n=input(['Enter m(1,',num2str(i),')']);
    m(1,i)=n;
end
[c]=find(m>10);
value=m(1,c);
disp(['The numbers greater than 10 are :',
num2str(value)])
disp(['Their location:', num2str(c)])
```

٤-٤ اكتب برنامجا يقوم بإدخال مصفوفتين ذات بعدين من النوع الصحيح، ثم يقوم بإجراء عملية الجمع علي المصفوفتين.

```
% Solution of exercise 4-4
A=input('Enter the first matrix')
B=input('Enter the second matrix')
if (size(A))==(size(B))
    C=A+B
else
    disp('Matrix dimentions must agree')
end
```

٤-٥ اكتب برنامجا لقراءة عناصر مصفوفة ذات بعدين ثم إيجاد وطباعة أصغر عنصر في المصفوفة.

```
% Solution of exercise 4-5
m=input('Enter a matrix')
m_min=min(min(m));
disp(['Te minimum of the matrix = ',num2str(m_min)])
```

٤-٦ اكتب برنامجا لقراءة عناصر مصفوفة ذات بعدين، ثم إيجاد حاصل ضرب عناصر القطر الرئيس للمصفوفة وطباعة أكبر عنصر في المصفوفة.

```
% Solution of exercise 4-6
A=input('Enter a square matrix')
[r,c]=size(A);
A_diag=diag(A)
A_max=max(max(A));
m=1;
for i=1:r
    m=m*A_diag(i);
end
diagonal_mul=m;
A_max=max(max(A));
disp(['Multiplication of elements of the main diagonal =
',num2str(m)])
disp(['Te maximum of the matrix = ',num2str(A_max)])
```

٤-٧ اكتب برنامجا لحل المعادلات الآتية باستخدام المصفوفات:

$$2x-3y+4z=5$$
$$x+y+4z=10$$
$$3x+4y-2z=0$$

```
% Solution of exercise 4-7
C=[2,-3,4;1,1,4;3,4,-2]
B=[5;10;0]
A=C\B;
x=A(1)
y=A(2)
z=A(3)
```

الفصل الخامس

أساسيات الرسم في ماتلاب

(٥,١) مقدمة

يحتوي ماتلاب على العديد من دوال ووسائل الرسم ثنائي وثلاثي الأبعاد التي تستخدم في كل التطبيقات بلا استثناء سواء التطبيقات الهندسية منها أو غير الهندسية ؛ فالكثير منا يريد دائما رؤية نتيجة تنفيذ برنامجه أو الخواريزم الخاص به في صورة معبرة ثنائية أو ثلاثية الأبعاد. ونحن سنغطى في هذا الفصل الكثير من الدوال التي تستخدم لرسم الدوال في نظم مختلفة للمحاور أو الرسم ثلاثي الأبعاد. كذلك سنعرض كيفية تعامل الماتلاب مع الرسم من خلال نوافذ الأشكال، والأنماط المختلفة للرسم في الماتلاب. نذكر هنا أن ماتلاب لديه وسائل مساعدة help غاية في السهولة والاكتمال، فإذا كنت تريد المزيد من المعلومات عن أي دالة فعليك طلب المساعدة عن هذه الدالة من نافذة الأوامر command window.

(٥,٢) أساسيات الرسم ثنائي الأبعاد

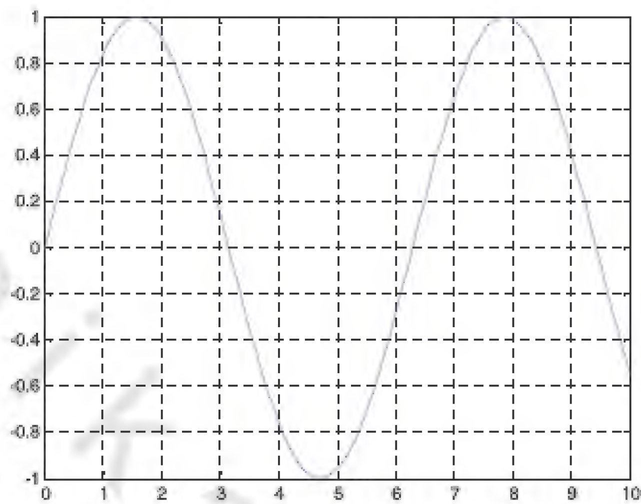
ماتلاب غني جدا بدوال الرسم التي تمكنك من رسم أي دالة في بعدين أو ثلاثة. من دوال الرسم ثنائي الأبعاد الدالة plot(x,y) التي ترسم المتغير أو المتجه y مع المتغير أو المتجه x. يجب ملاحظة أن قيم المتغير الأول (x) تمثل على المحور الأفقي بينما يتم تمثيل قيم المتغير الثاني (y) على المحور الرأسي. وعلى ذلك يجب الأخذ في الاعتبار الترتيب لأن ناتج تنفيذ الرسم plot(x,y) يختلف عن plot(y, x). كمثال على ذلك سنرسم الدالة y=sin(x) كما في الأوامر التالية وكما هو موضح في شكل (٥,١) :

```
>> x=0:0.1:10;
>> y=sin(x);
>> plot(x,y), grid
```

الأمر x=0:0.1:10 سيجعل x تتغير من صفر حتى ١٠ بالتقدير الدائري والفرق بين كل نقطة والثانية هو ٠,١ ، والأمر y=sin(x) سيحسب قيمة y المقابلة لكل x، ثم الأمر plot(x,y) الذي يرسم y مع x، ثم الأمر grid الذي يرسم الشبكة التي نراها في الشكل. حاول تجربة الأمر plot(y, x) ولاحظ الفرق.

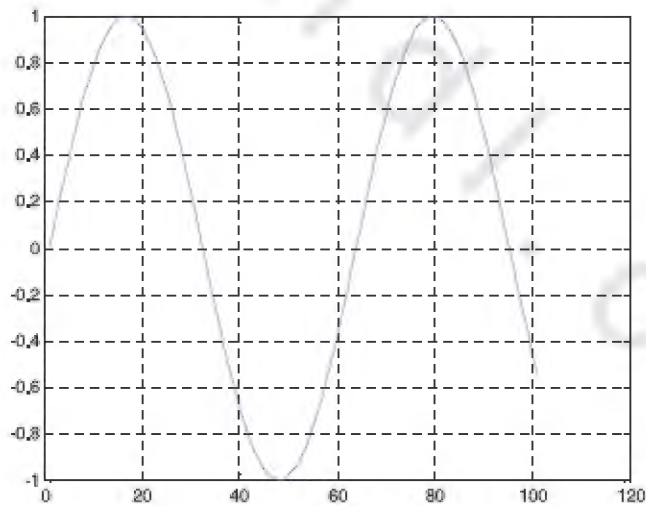
دالة الرسم plot() لها متغيرات كثيرة جدا سنذكر منها ما يلي :

الدالة plot(y) سترسم المتغير y مع فهرسه أي بترتيب نقط هذا المتغير أي عند النقطة ١ و٢ و٣ وهكذا ؛ لذلك فإننا لو نفذنا ذلك على الرسم السابق سنحصل على الشكل (٥,٢) حيث نلاحظ أن المحور x هنا يتغير من صفر حتى ١٢٠ حيث هناك ١٠٠ نقطة للمتغير x، بينما في شكل (٥,١) فقد تم رسم المتغير y مع قيم x التي تنتهي عند القيمة ١٠.



شكل (٥،١). الدالة y=sin(x) باستخدام الأمر plot(x,y).

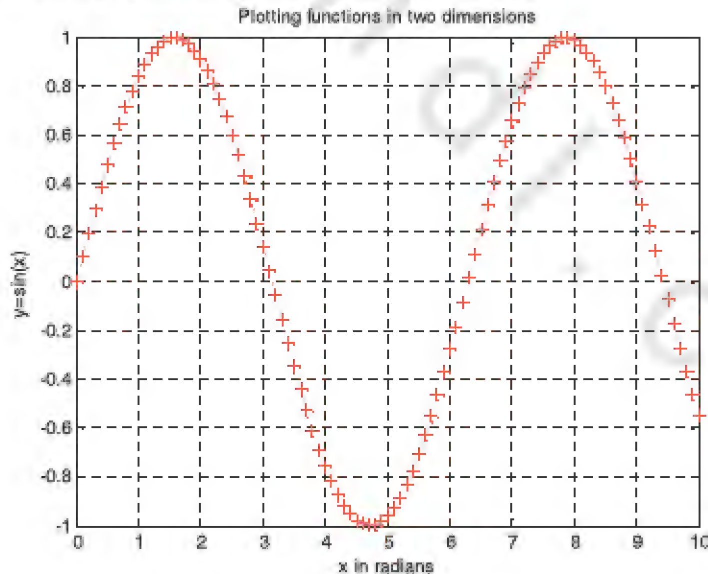
>> plot(y) , grid



شكل (٥،٢). رسم الدالة y=sin(x) باستخدام الأمر plot(y).

يمكن التحكم في لون المنحنى وإضافة أسماء للمحاور وعنوان للشكل كما في شكل (٥,٣). في الأمر plot() التالي تم إضافة علامتي تنصيص يوضع بينها اختيار اللون بالحرف r للون الأحمر و '+' لتوقيع نقط الرسم أو جعل نقطة الرسم هي الحرف +، و ':' لجعل المنحنى منقطاً بدلا من منحنى متصل. بعد ذلك تم إضافة اسم للمحور x بالأمر xlabel()، وإضافة اسم للمحور y بالأمر ylabel()، ثم إضافة عنوان للشكل بالأمر title. لاحظ أن إضافة المسميات السابقة لابد أن تكون بين علامتي تنصيص. (في الشكل (٥,٣) لون المنحنى أحمر ولكنه طبعا سيظهر أسود عند طباعة الكتاب ولذلك نوصى الدارس بكتابة هذه الأوامر وملاحظة الخرج) .

```
>> plot(x,y,'r+:')
>> grid
>> xlabel('x in radians')
>> ylabel('y=sin(x)')
>> title('Plotting functions in two dimensions')
```



شكل (٥,٣). تسمية المحاور وإضافة عنوان للرسم.

جدول (٥،١) يبين الحروف المستخدمة لاختيار اللون ونمط المنحنى فحاول تجربة هذه المتغيرات المختلفة.

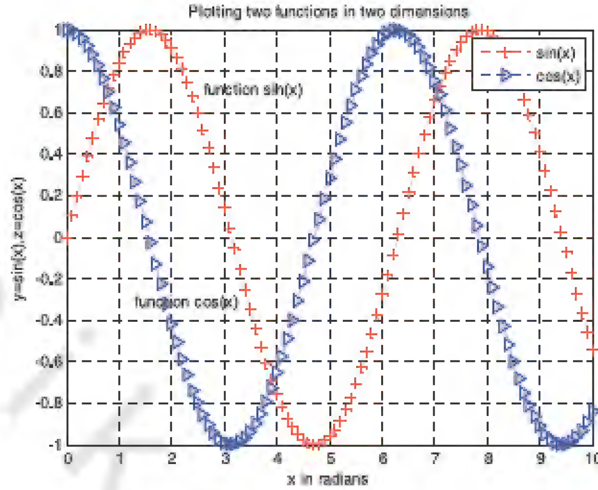
جدول (٥،١). لون ونمط المنحنى ونمط التنقيط مع الدالة plot().

نمط المنحنى		نمط التنقيط		لون المنحنى	
خط مستمر solid	-	نقطة point	.	أزرق blue	b
خط منقط dotted	:	دائرة circle	o	أخضر green	g
شحطة ونقطةdashdot	-.	علامة x x mark	x	أحمر red	r
خط متقطع dashed	--	علامة الجمع plus	+	cyan أزرق مائل للخضرة	c
بدون لون no line	none	نجمة star	*	magenta أحمر بنفسجي	m
		مربع square	s	أصفر yellow	y
		معين diamond	d	أسود black	k
		مثلث (أسفل) triangle(down)	v	أبيض white	w
		مثلث (أعلى)triangle(up)	^		
		مثلث (يسار) triangle(left)	<		
		مثلث(يمين) triangle(right)	>		
		نجمة خماسية pentagram	p		
		نجمة سداسية hexagram	h		

يمكن رسم أكثر من منحنى بالأمر plot() الذي سنستخدمه لرسم الدالة sin(x) و cos(x) كما سنرى في شكل (٥،٤) حيث في هذا الشكل تم رسم منحنيين في نفس الشكل. وتم إضافة مفتاح legend لهذا الشكل نميز به كل منحنى عن الآخر كما في

الشكل حيث يوضح المفتاح أن المنحنى المنقط بعلامة زائد يمثل المنحنى sin(x)، بينما المنحنى المنقط بالمثلثات هو منحنى الدالة cos(x). هناك متغيرات كثيرة عن مفتاح الشكل من حيث مكانه وحجمه ونوع الخط فيه يمكنك معرفتها باختصار بكتابة الأمر help legend أو معرفتها بالتفصيل بالنقر على قائمة help ثم ابحث عن legend من مكان البحث حيث سيظهر لك معلومات تفصيلية بأمثلة توضيحية عن الأمر legend. لاحظ أنه بالنقر مرتين على المفتاح يمكنك تعديل النصوص المكتوبة فيه مباشرة. كما أنه يمكنك الوقوف على المفتاح والضغط عليه بالفأرة مرتين مع السحب والإسقاط في أي مكان على الشكل الناتج من ماتلاب لتغيير مكان المفتاح، فحاول ذلك. حاول تنفيذ البرنامج التالي للحصول على شكل (٥,٤).

```
>> x=0:0.1:10;
>> y=sin(x);
>> z=cos(x);
>> plot(x,y,'r+:',x,z,'b>:')
>> grid
>> title('Plotting two functions in two dimensions')
>> ylabel('y=sin(x),z=cos(x)')
>> xlabel('x in radians')
>> legend('sin(x)','cos(x)')
>> gtext('function sin(x)')
>> gtext('function cos(x)')
```



شكل (٥,٤). رسم أكثر من منحنى باستخدام الأمر plot().

يمكنك أيضا كتابة أي نص بجوار أي منحنى في الشكل وفي أي مكان باستخدام الدالة gtext(). في شكل (٥,٤) كتبنا النص function sin(x) والنص function cos(x) باستخدام هذه الدالة. عندما يبدأ ماتلاب بتنفيذ هذه الدالة سيظهر أمامك على الشكل خطان متقاطعان ونقطة تقاطع هذان الخطان تتحرك مع الفأرة، اختر المكان الذي تريد كتابة النص عنده، وقف بالفأرة عند هذا المكان ثم انقر الفأرة، ستجد أن النص الموجود في الدالة gtext() بين علامتي التنصيص قد تم إسقاطه في نفس المكان تماما كما في شكل (٥,٤). هناك بعض المتغيرات المفيدة للدالة gtext() يمكنك الاطلاع عليها بطلب المساعدة من ماتلاب بالأمر help gtext والذي نعرض جزءا منه كما يلي:

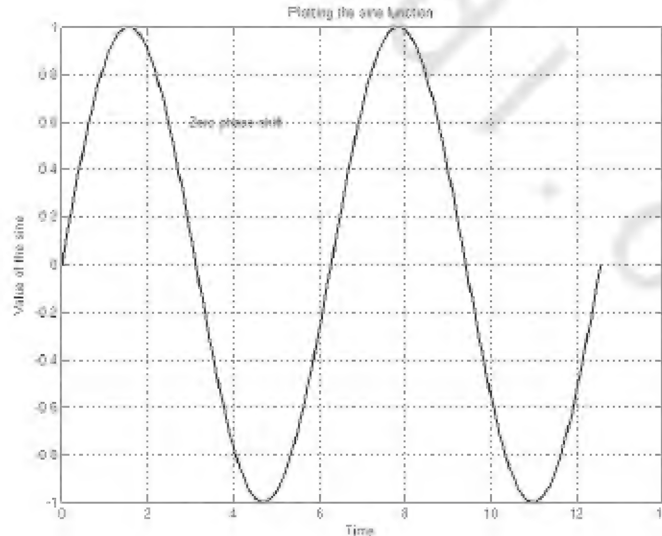
```
>> help gtext
 GTEXT  Place text with mouse.
   GTEXT('string') displays the graph window, puts up a
   cross-hair, and waits for a mouse button or keyboard key to be
   pressed.  The cross-hair can be positioned with the mouse (or
   with the arrow keys on some computers).  Pressing a mouse button
   or any key writes the text string onto the graph at the selected
   location.
```

الطريقة السابقة لإضافة النص على الرسم تسمى الطريقة التفاعلية، حيث تقوم أنت تفاعليا باختيار النقطة على الرسم التي ستضع عندها النص ثم تضرب enter فيظهر النص كما سبق وأوضحنا. يمكن إضافة هذه النصوص على الرسم بتحديد المكان الذي ستضع عنده هذا النص كما يلي :

text(x,y,'text');

حيث يتم وضع النص 'text' عند النقطة x و y على الرسم. شكل (٥,٥) يبين الدالة sin بعد إضافة نص عليها باستخدام البرنامج التالي.

```
%Plotting 1
x = 0:pi/40:4*pi;
plot(x, sin(x),'k','LineWidth',2);
xlabel('Time');
ylabel('Value of the sine');
title('Plotting the sine function');
text(3,0.6,'Zero phase shift');
grid
```



شكل (٥,٥). إضافة نص على الرسم باستخدام الأمر text.

في العادة يتولى ماتلاب بنفسه ضبط المحاور على ضوء البيانات المتاحة للمتغيرات. في نفس الوقت فإن ماتلاب أعطى الحرية للمستخدم أن يلغى ذلك ويقدم هو القيمة العظمى والصغرى للمحورين بالأمر التالي:

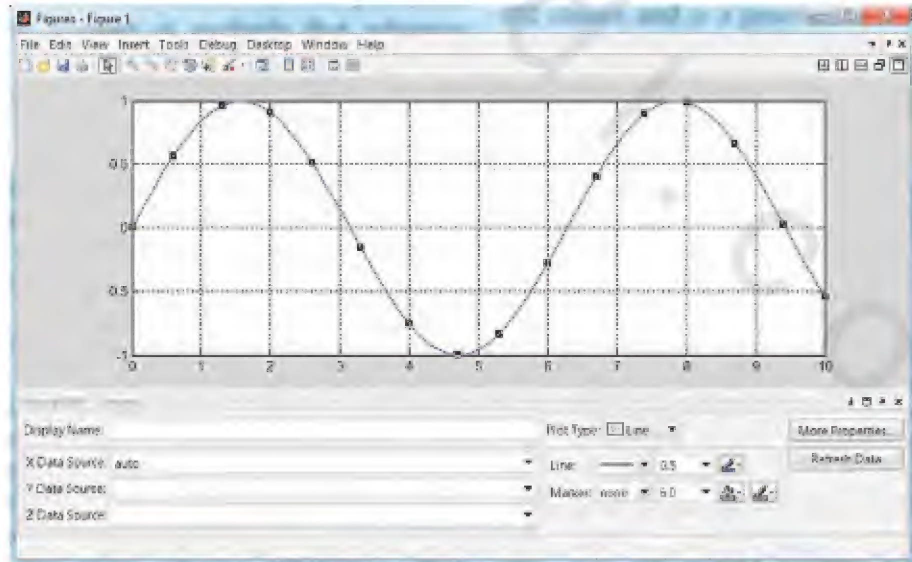
```
axis ([xmin, xmax, ymin, ymax]);
```

حيث xmin و xmax هي القيمة الصغرى والعظمى للمحور x. وكذلك ymin و ymax هي القيمة الصغرى والعظمى للمحور y.

يمكن إرجاع التحكم في المحاور إلى ماتلاب مرة ثانية عن طريق الأمر:

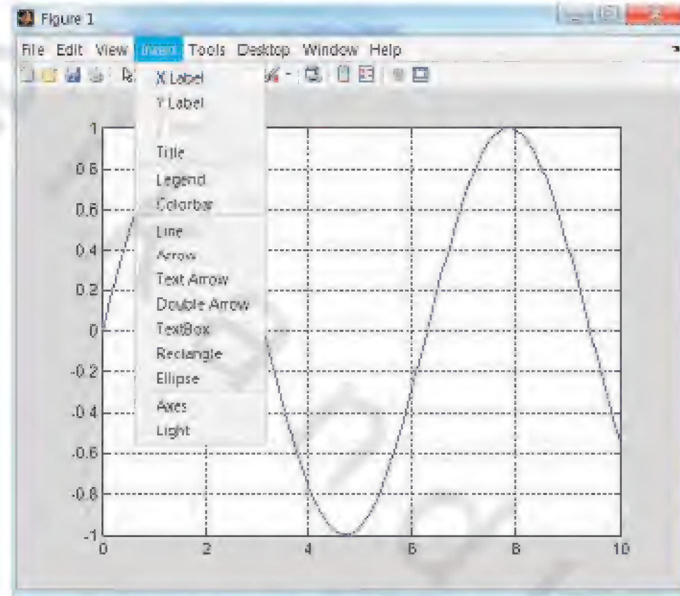
```
axis auto
```

يمكن استخدام محرر الرسم لتحسين أي شكل في الماتلاب عن طريق فتح قائمة Tools من شريط القوائم ثم اختيارEdit plot وبالنقر مرتين على الشكل يتم فتح المحرر كما في شكل (٥،٦). هنا يمكن تغيير لون المنحنى ونمط الخط وكذلك إضافة أسماء المحاور وعنوان مفتاح للشكل والكثير من الإضافات التي من شأنها تحسين الشكل.



شكل (٥،٦) فتح محرر الرسم.

حاول الحصول على الأشكال السابقة (من شكل (٥,٣) إلى شكل (٥,٥)) عن طريق تجربة محرر الرسم مع الشكل (٥,١). عن طريق فتح قائمة Insert من شريط القوائم يمكن أيضا إضافة المحاور أو عنوان أو حتى خط أو سهم كما موضح في شكل (٥,٧).



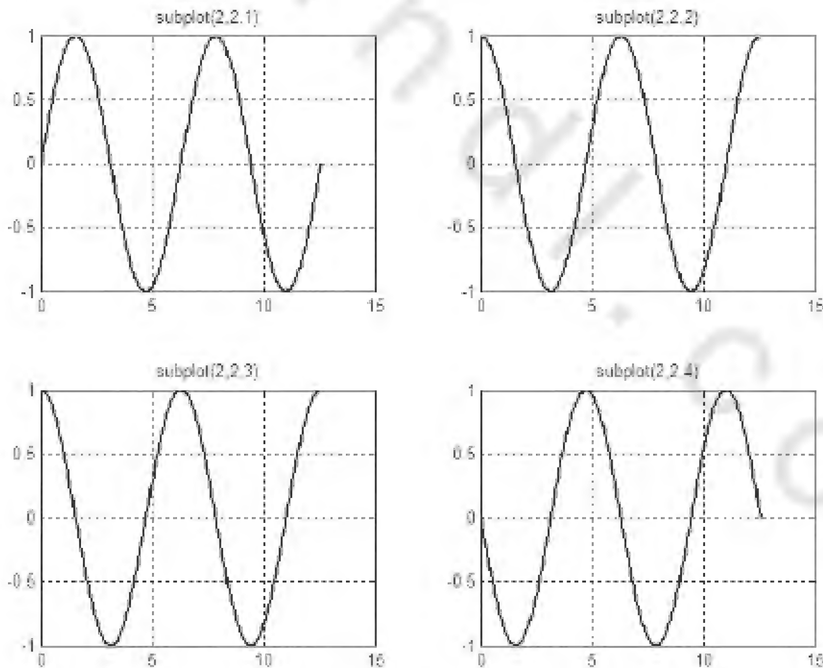
شكل (٥,٧) قائمة Insert.

يمكن الحصول على نسخة من أي شكل عن طريق فتح قائمة Edit من شريط القوائم ثم اختيار Copy figure ثم لصق هذه النسخة في أي ملف وورد أو بوربوينت.

(٥,٢,١) وضع أكثر من شكل في نافذة الرسم باستخدام الأمر subplot

الأمر subplot (m,n,p) يقسم نافذة الرسم إلى مصفوفة mxn من الأشكال الصغيرة والحرف p يرمز لرقم الشكل في النافذة بحيث يتم ترقيم هذه الأشكال من أول صف ثم الثاني ثم الثالث وهكذا ويكون العد من اليسار لليمين. البرنامج التالي يقسم نافذة الرسم إلى أربعة أشكال، لاحظ كيف يتم ترتيب هذه الأشكال كما في شكل (٥,٨):

```
%Training with subplot
x = 0:pi/40:4*pi;
subplot(2,2,1)
plot(x, sin(x),'k','LineWidth',2);
title('subplot(2,2,1)')
grid
subplot(2,2,2);
plot(x, cos(x),'k','LineWidth',2);
title('subplot(2,2,2)');grid;
subplot(2,2,3)
plot(x, sin(x+pi/2),'k','LineWidth',2);
title('subplot(2,2,3)'); grid;
subplot(2,2,4);
plot(x, cos(x+pi/2),'k','LineWidth',2);
title('subplot(2,2,4)'); grid;
```

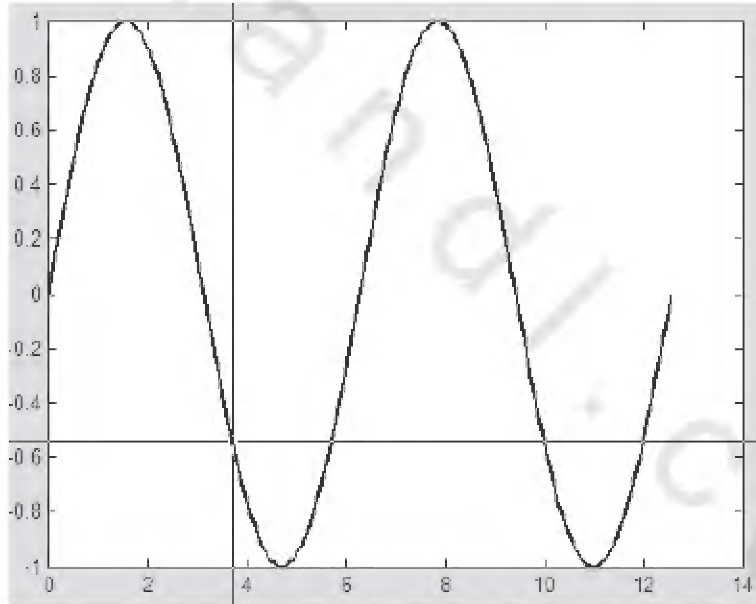


شكل (٥.٨). إظهار أكثر من شكل في نفس نافذة الرسم.

عندما يكون لديك شكل معين مرسوم على نافذة الرسم في ماتلاب وتريد قراءة إحداثيات نقطة من نقاط هذه الدالة المرسومة فإن ذلك يكون ممكنا عن طريق الدالة :

```
[x,y]=ginput
```

حيث بمجرد تنفيذ هذه الدالة سيظهر أمامك خطان متعامدان متلازمان لدليل الكتابة cursor ويتحركان معه بحيث عند النقر على أي نقطة من نقاط الشكل يتم تخزين إحداثيات هذه النقطة في المتغيرين x و y بعد قراءة أي عدد من النقاط التي تريدها اضرب enter حيث ستظهر لك فورا الإحداثيات المخزنة في المتغيرين x و y. شكل (٥,٩) يوضح دليل اختيار النقاط المراد قراءة إحداثياتها.



شكل (٥,٩). دليل اختيار النقط المراد قراءة إحداثياتها.

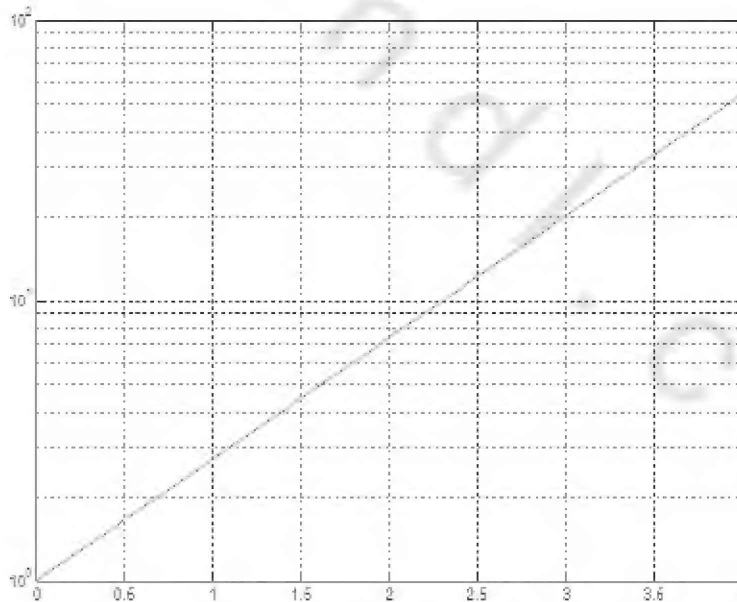
الدالة [x,y]=ginput(n) ستقرأ إحداثيات عدد n من النقاط وتظهر إحداثياتها في المتجه x و y.

في الكثير من التطبيقات وبالذات في تطبيقات التحكم ومعالجة الإشارات نحتاج لرسم دالة على محاور لوغاريتمية للأساس ١٠. يمكن عمل ذلك باستخدام الدالة semilogy كما في المثال التالي :

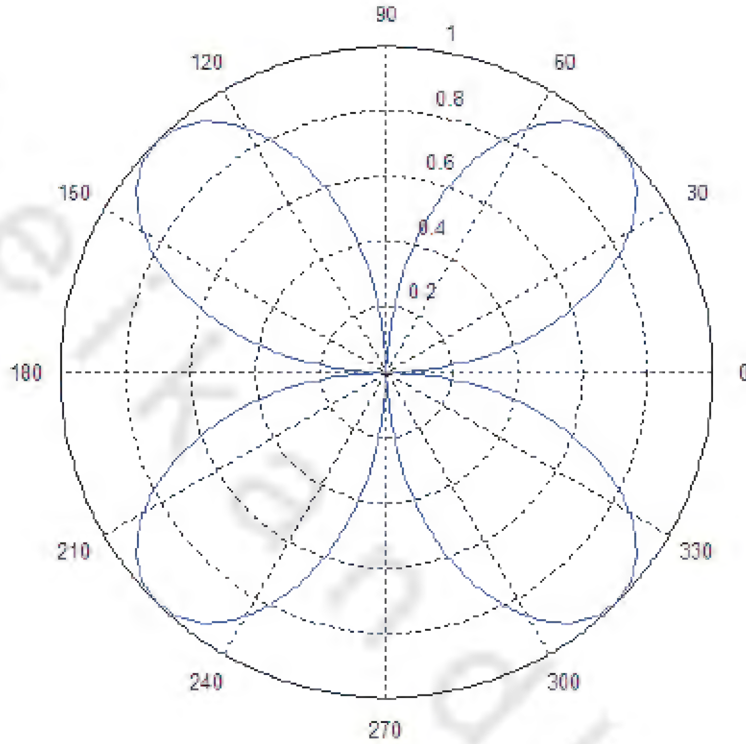
```
>> x=0:.01:4;
>> semilogy(x, exp(x)), grid
```

تنفيذ هذين الأمرين كما في الشكل (٥،١٠) يبين نتيجة الدالة الأسية exp(x) وقد ظهرت كخط مستقيم لأنها مرسومة على المحور y اللوغاريتمي. لاحظ أن هذه الدالة وكذلك المتغيرات المختلفة لها تعمل بنفس طريقة الدالة plot التي سبق شرحها.

هناك أيضا الدالة semilogx حيث في هذه الحالة سيكون المحور الأفقي هو المحور اللوغاريتمي. هناك أيضا الدالة loglog التي ترسم على محورين لوغاريتميين الأفقي والرأسي.



شكل (٥،١٠). الرسم على محاور لوغاريتمية.



شكل (٥,١١). الرسم على المحاور القطبية.

(٥,٢,٢) الرسم على إحداثيات قطبية Polar plot

أي نقطة في الإحداثيات الكارتيزية x و y يمكن تمثيلها على الإحداثيات القطبية بالعلاقات التالية :

$x=r\cos(\theta)$, $y=r\sin(\theta)$

حيث θ تتغير من صفر حتى ٢π أو ٣٦٠ درجة. كمثال على ذلك البرنامج التالي الذي يعطي الرسم القطبي الموضح في شكل (٥,١١).

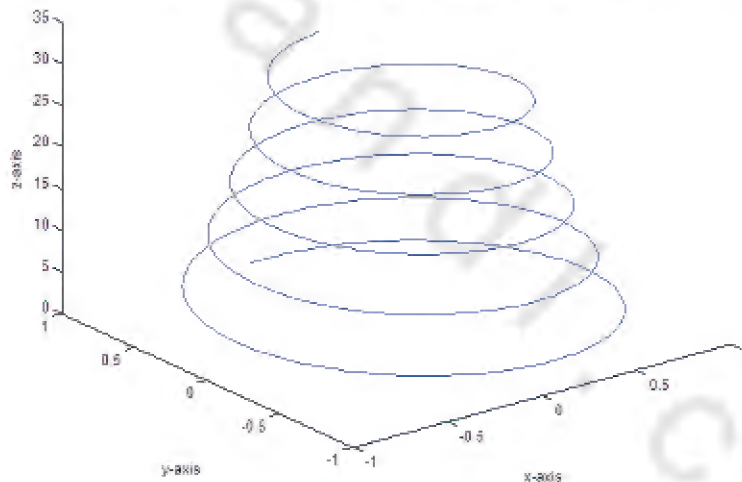
```
>> x = 0:pi/40:2*pi;
>> polar(x, sin(2*x)),grid
```

(٥,٣) الرسم ثلاثي الأبعاد

يحتوي ماتلاب على العديد من دوال الرسم ثلاثي الأبعاد التي سنحاول التعرف عليها في هذا الجزء. أول هذه الدوال الدالة plot3() التي تقوم بما تقوم به الدالة plot() في الرسم الثنائي الأبعاد، ولكن هذه المرة في المستوى ثلاثي الأبعاد. البرنامج التالي يوضح ذلك:

```
%training with 3D plot
t = 0:pi/50:10*pi;
plot3(exp(-0.02*t).*sin(t), exp(-0.02*t).*cos(t),t);
xlabel('x-axis'); ylabel('y-axis'); zlabel('z-axis');
```

نتيجة تنفيذ هذا البرنامج موضحة في شكل (٥,١٢).



شكل (٥,١٢) الرسم ثلاثي الأبعاد.

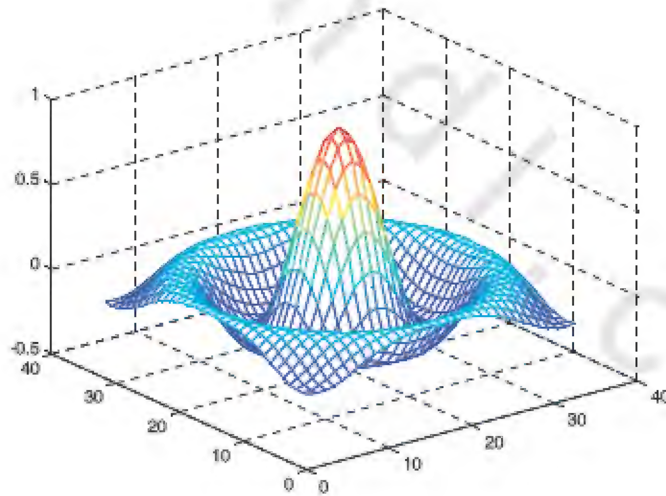
الدالة comet3() تعمل نفس عمل الدالة plot3() ولكن بصورة حركية، حيث ترى نقطة الرسم أثناء رسمها للمنحنى الثلاثي الأبعاد وهي تتحرك من بداية الرسم إلى نهايته: حاول نفس المنحنى السابق ولكن هذه المرة باستخدام الأمر comet().

```
%training with 3D comet
t = 0:pi/50:10*pi;
comet3(exp(-0.02*t).*sin(t), exp(-0.02*t).*cos(t),t);
```

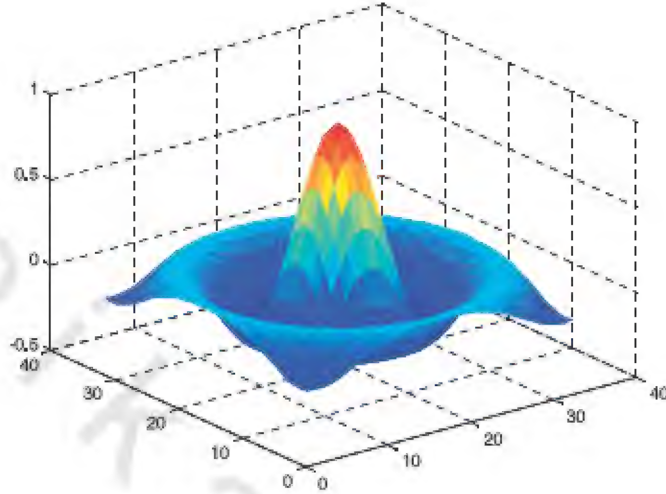
يمكن رسم القبعة المكسيكية باستخدام الدالة ()meshgrid التي تشبه إلى حد كبير الدالة ()plot3 أو ()comet3 كالتالي :

```
>> [x y ] = meshgrid(-8 : 0.5 : 8);
>> r = sqrt(x.^2 + y.^2) + eps;
>> z = sin(r) ./ r; mesh(z);
```

شكل (٥،١٣) يبين القبعة المكسيكية الشهيرة، وقد تم رسمها باستخدام الأربعة أوامر السابقة فقط مما يبين سهولة الرسم ثلاثي الأبعاد في ماتلاب وروعته. يمكنك الحصول على شكل أجمل من خلال إضافة ظلال لهذا الشكل الثلاثي بالأمر surf(z) وكما هو موضح في شكل (٥،١٤).



شكل (٥،١٣). القبعة المكسيكية كشكل ثلاثي الأبعاد.



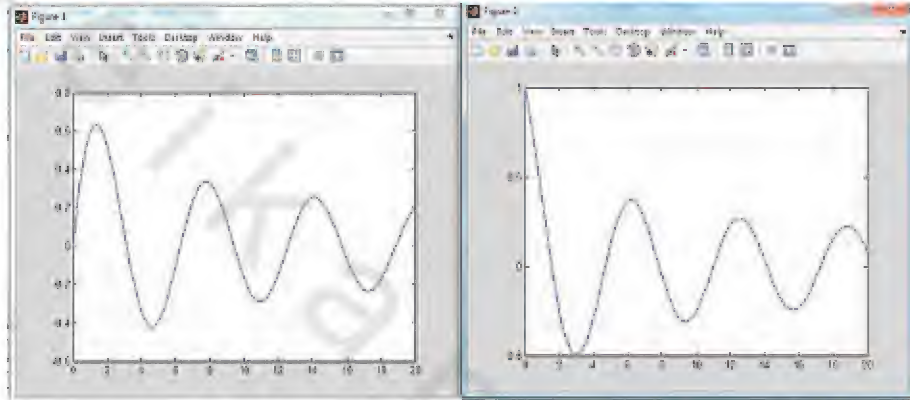
شكل (٥،١٤). القبعة المكسيكية بعد إضافة ظلال لها.

نكتفي بهذا الكم من أوامر الرسم، وبالطبع سنرى في الأجزاء القادمة أوامر وبرامج تنتج عنها طرق متقدمة تعرض أنماطا أخرى من الرسم.

(٥،٤) التعامل من خلال نوافذ الشكل

يعرض ماتلاب جميع رسوماته من خلال نافذة شكل كالتي كنا نراها في جميع الرسومات السابقة. إنك حين تقوم بتنفيذ أي أمر من أوامر الرسم يتم تنفيذ هذا الأمر داخل نافذة شكلية، ولكن في نفس الوقت يمكنك طلب فتح نافذة شكلية جديدة لتعرض فيها رسما جديدا دون أن تعرض هذا الرسم في نفس الشكل كما كنا نفعل مسبقا فربما يحتاج الأمر لذلك في بعض التطبيقات كما في البرنامج التالي الذي رسم الدالة sin() والدالة cos() كما في شكل (٥،١٥) كل منهما في نافذة شكل منفصلة باستخدام الأمر figure الذي فتح نافذة جديدة رسمنا فيها الدالة cos().

```
>> x=[0:0.2:20];
>> y=sin(x)./sqrt(x+1);
>> z=cos(x)./sqrt(x+1);
>> plot(x,y)
>> figure
>> plot(x,z)
```



شكل (٥,١٥). عرض رسمين كل منهما في شكل منفصل باستخدام الأمر figure.

فيما سبق رأينا كيف نعدل من مظهر أي رسم عن طريق جعل الخط أكثر سمكا أو تغيير اللون أو تغيير شكل نقاط الرسم أو وضع نصوص معينة على الرسم عن طريق أوامر منفصلة أو عن طريق بعض المتغيرات في الأمر plot() نفسه. سنرى هنا كيف يمكن تغيير مظهر هذا الرسم عن طريق أوامر خارجية ومن سطح المكتب دون الدخول في برنامج الرسم أو تعديل أمر الرسم وذلك بالاستفادة من الخاصية الفريدة في ماتلاب، وهي أنه ينظر لنافذة الرسم على أنها هدف أو شيء منفصل، وهذا الشكل يمكن التعامل معه من خلال مسمى handle وهو في العادة يكون رقم هذه النافذة الشكلية. فمثلا إذا نقرنا على النافذة الشكلية figure (1) لتكون هي النافذة الشكلية الفعالة ثم نفذنا الأمر:

```
>> hf=gcf
hf =
    1
```

فإن hf في هذه الحالة تكون هي مسمى هذه النافذة الشكلية. كذلك الأمر :

```
>> hf=gcf
hf =
    2
```

الذي يعطي مسمى النافذة الثانية figure (2) بعد النقر عليها. نلاحظ أن مسمى النافذة يكون هو رقم النافذة الذي يظهر في شريط العنوان لهذه النافذة. بمعرفة هذا المسمى لأي نافذة يمكن التغيير لكثير من الخواص الظاهرية لهذا الشكل. إن كل رسم داخل أي نافذة يكون له مسمى آخر، ويمكن معرفة هذا المسمى بالنقر على هذا الرسم. فمثلا إذا نقرنا على المنحنى cos() نفسه في النافذة الشكلية ٢ ثم كتبنا الأمر التالي :

```
>> ho=gco
ho =
  345.0042
```

معنى ذلك أن منحنى الـ cos في النافذة الشكلية ٢ هو ٣٤٥,٠٠٤٢ وهذا المسمى من قبل ماتلاب نفسه. بمعرفة هذا المسمى يمكن الآن من على سطح المكتب تغيير الشكل الظاهري لهذا المنحنى كما في الأمر التالي :

```
>> set(ho,'linewidth',4)
```

والذي ستكون نتيجته هي تغيير سمك خط المنحنى إلى ٤ أضعافه. كما يمكن تغيير شكل المنحنى باستخدام الأمر :

```
>> set(ho,'linestyle','-.')
```

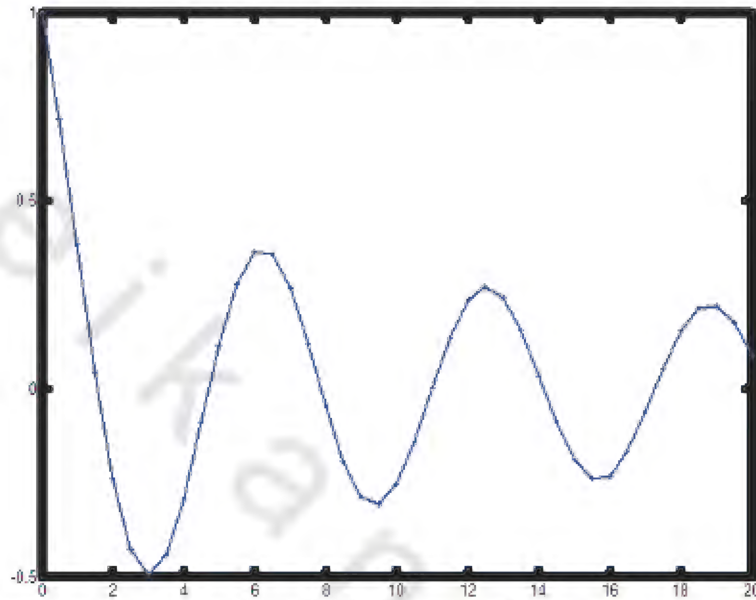
ليكون الشكل المنقط كما في شكل (٥,١٦)، كما يمكن تغيير شكل سمك المحاور بمعرفة مسمى محاور هذا الشكل بالأمر :

```
>> ha=gca
ha =
  344.0051
```

ثم تغيير سمك المحاور بالأمر :

```
>> set(ha,'linewidth',6)
```

هذه الأوامر مجتمعة ستجعل figure (2) يبدو كما في شكل (٥,١٦).

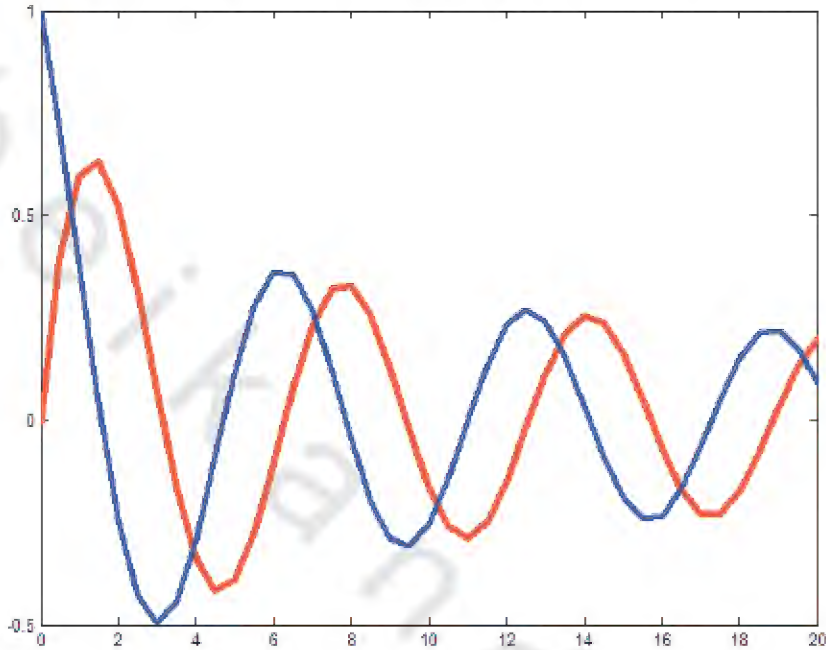


شكل (٥,١٦). بعد تعديل الكثير من مظاهرة من سطح المكتب باستخدام أوامر من خارج الأمر plot() نفسه.

هناك الكثير من الخواص التي يمكن تغييرها باستخدام مسمى الرسم object handle داخل النافذة الشكلية. يمكن الحصول على كل هذه الخواص، وهي تصل إلى ما يقرب من الثلاثين خاصية، والتي سنعرض القليل منها بالأمر التالي فحاول تنفيذه ورؤية كل هذه الخواص:

```
>> get (ho)
       DisplayName: ''
        Annotation: [1x1 hg.Annotation]
             Color: [0 0 1]
         EraseMode: 'normal'
         LineStyle: '-.'
         LineWidth: 2
            Marker: '.'
        MarkerSize: 6
```

لاحظ أن ho في الأمر get هي مسمى منحنى الـ cos() في نافذة الشكل (٢).



شكل (٥,١٧). التعامل مع منحنيات النافذة الشكلية كل على حدة.

لاحظ أيضا أنه لتغيير شكل منحنى معين في نافذة شكلية ليس بالضرورة أن تكون النافذة الشكلية تحتوي على رسم أو منحنى واحد، ولكن من الممكن أن تحتوي النافذة الشكلية على أكثر من منحنى ثم بالنقر على هذا المنحنى لمعرفة مسماة handle من خلال الأمر ho=gco يمكن تغيير أشكال هذا المنحنى كما في الأوامر التالية التي تم استخدامها لتغيير شكل المنحنيات الموجودة في شكل (٥,١٧) كل على حدة:

```
>> ho=gco
ho =
  172.0063
```

تم ذلك بالنقر على منحنى الـ sin() للحصول على مسماه ثم غيرنا لونه إلى اللون الأحمر بالأمر التالي، ثم تغيير سمكه بالأمر التالي له:

```
>> set(gco,'color','r')
>> set(gco,'linewidth',4)
>> bo=gco
bo =
  173.0059
```

تم ذلك أيضا بالنقر على منحنى الـ cos() ثم تغيير سمكه وشكله بالأمرين التاليين:

```
>> set(gco,'linewidth',3)
>> set(gco,'linestyle','-.')
```

(٥,٥) التعامل مع الرسم من خلال النوافذ الشكلية مباشرة

كما ذكرنا فإن أي رسم في الماتلاب يظهر من خلال نافذة شكلية، وهذه النافذة الشكلية تحتوي ثلاثة شرائط في قمتها تتوافق مع شرائط نوافذ ميكروسوفت المعروفة كالموضحة في شكل (٥,١٨). نلاحظ وجود شريط للعنوان باللون الأزرق غالبا يحتوي على اسم الشكل figure (1) مثلا، ثم شريط للقوائم المتوافقة تماما مع قوائم نوافذ ميكروسوفت مثل قائمة الملفات File وقائمة التحرير Edit وغيرها، ثم شريط الأيقونات الذي يحتوي أيقونات خاصة بالكثير من العمليات التي يمكن إجراؤها على محتويات هذه النافذة.



شكل (٥,١٨). شرائط النافذة الشكلية.

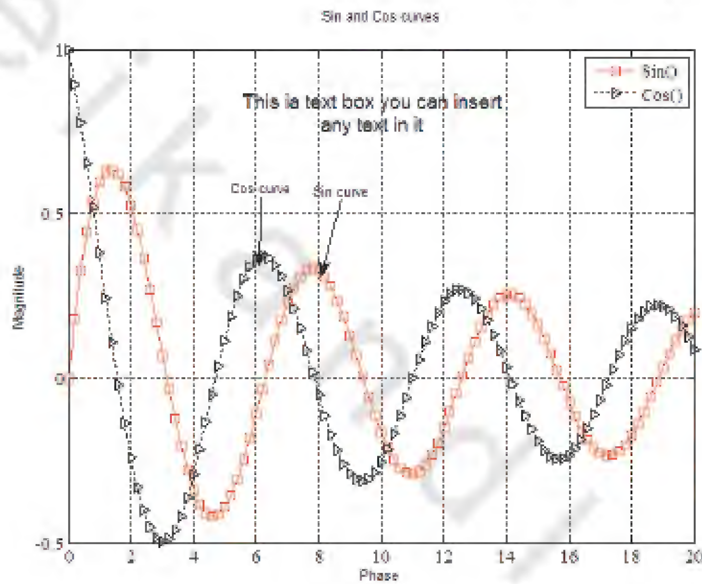
شكل (٥،١٩) يبين شريط الأيقونات مع مسمى كل أيقونة من هذه الأيقونات.



شكل (٥،١٩). شريط الأيقونات ومحتوياته.

سنبدأ هنا بعرض عمل أيقونة عرض/إخفاء أدوات عرض الرسم، حيث بالنقر على هذه الأيقونة ستظهر أمامك نافذة الرسم وتحتها مساحة تحرير خواص الشكل، حيث من هذه المساحة يمكن عمل الكثير من أغراض تحرير محتويات الشكل. سنعرض هنا الشكل الذي يحتوي على دالتي الـ sin() والـ cos() لنعرض عليهما نتيجة هذا التحرير. في شكل (٥،٢٠) سننقر على منحنى الـ sin() مرتين، حيث ستظهر أمامك خواص لهذا المنحنى، والتي منها يمكن أن نختار شكل الخط من خلال الخاصية line ثم سمك هذا الخط من الرقم المجاور ثم لون الخط أو المنحنى من المربع المجاور، يمكن كذلك اختيار شكل النقاط marker وسمك هذه النقاط ولونها من المربعات المجاورة. بعد ذلك انقر على المنحنى الثاني منحنى الـ cos() واختر له هذه الخواص. بعد ذلك ننقر على أي نقطة أخرى في الشكل بعيدا عن المنحنيات، حيث ستظهر لك مجموعة أخرى من الخواص والتي منها عنوان لهذا الشكل title حيث هناك مربع يمكنك أن تضع فيه أي عنوان لهذا الشكل، وقد كتبنا Sin and Cos curves كعنوان لهذا الشكل. يمكنك منها أن تختار لوناً للشكل ولوناً للمنحنيات ثم اختيار عنوان لكل محور، حيث اخترنا عنوان المحور x أنه هو Phase وعنوان المحور y هو Magnitude. يمكن تغيير حجم ولون

الكتابة على كل منحنى أيضا، كما يمكن تغيير حدود كل محور في حالة إذا كانت الحدود التلقائية غير مرضية. يمكن أيضا إضافة مفتاح أو Legend في أعلى يمين الشكل، حيث يتم ذلك من خلال أيقونة وضع مفتاح الرسم . هذه هي أهم الإضافات التي يمكن عملها من خلال محرر نافذة الشكل، والتي أوضحناها في شكل (٥،٢٠) حاول التجربة مع باقي هذه الخواص.



شكل (٥،٢٠). تحرير الشكل من خلال محرر نافذة الشكل.

هناك الكثير في قائمة الإدخال Insert فحاول استعمالها وتجربتها لعمل مربع نص يمكن إضافة أي نص فيه، وكذلك إضافة أسهم مع نص يشير إلى أي منحنى في الشكل. من قائمة الإظهار View يمكنك إظهار شريط المشاهدة بالكاميرا وشريط تحرير الشكل، والتي من خلال كل منهما يمكن عمل الكثير بالذات مع الرسومات الثلاثية الأبعاد والتي سنتركها للمستخدم لأنها ربما تكون خارج أهداف هذا الكتاب أن ندخل في تفاصيل هذه الشرائط.



شكل (٥,٢١). دوال رسم الخطوط أو المنحنيات.

(٥,٦) أنواع الرسم المختلفة في ماتلاب

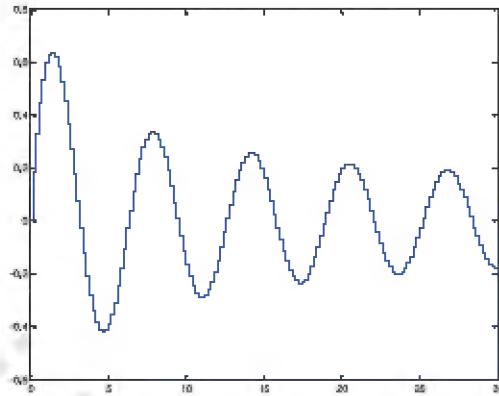
هناك الكثير من دوال الرسم التي يمكن الحصول منها على أنماط مختلفة من الرسم. نقدم في هذا الجزء هذه الدوال مع نبذة مختصرة لبعضها الذي لم يتم ذكره أو استخدامه من قبل. شكل (٥,٢١) يقدم أول نوعية من هذه الدوال وهي رسم المنحنيات أو الخطوط. لقد سبق وتعاملنا مع بعض هذه الدوال مثل الدوال plot, plotyy, loglog, semilogx, semilogy.

أما دالة السلالم فهي تقوم برسم المنحنى في صورة درجات سلم كما في البرنامج البسيط التالي وكما في شكل (٥,٢٢).

```
>> x=[0:0.2:30];
>> y=sin(x)./sqrt(x+1);
>> stairs(x,y)
```

يمكن بيان عمل الدالة contour من خلال رسم المعادلة $z=exp(-x^2-y^2)$ في المدى التالي $-2\leq x\leq 2$ و $-2\leq y\leq 3$. لاحظ أن الدالة contour ترسم ارتفاعات المتغير z فوق مستوى المتغيرين x و y. البرنامج التالي وشكل (٥,٢٣) يبينان هذه الدالة، ويمكنك طلب المساعدة عنها باستخدام طلب المساعدة help contour.

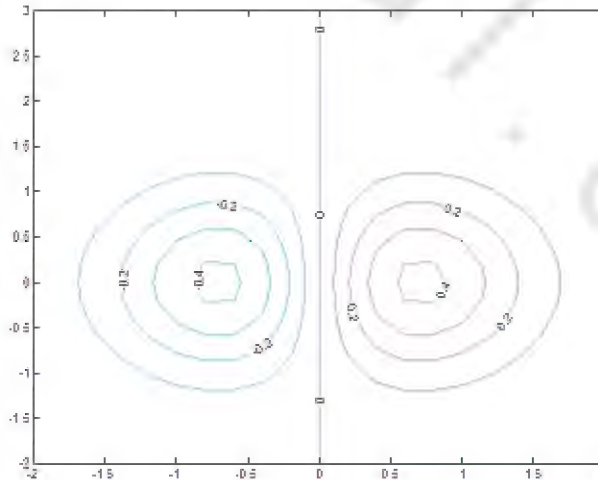
```
>> [x,y] = meshgrid(-2:.2:2,-2:.2:3);
>> z = x.*exp(-x.^2-y.^2);
>> [c,h] = contour(x,y,z);
>> set(h,'ShowText','on','TextStep',get(h,'LevelStep')*2)
>> colormap cool
```



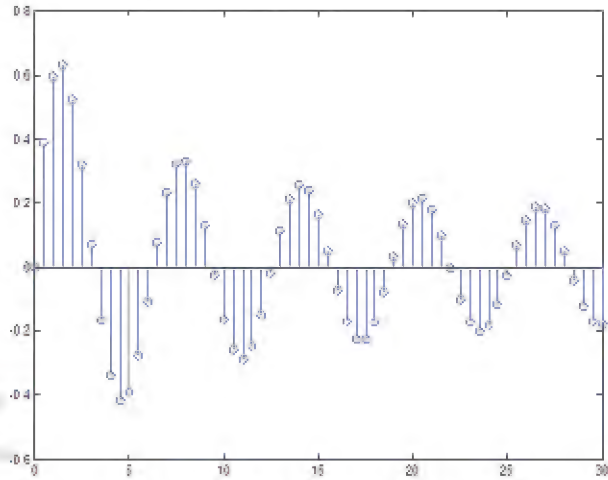
شكل (٥,٢٢) الرسم باستخدام الدالة stairs.

النمط الثاني من أنماط الرسم هو باستخدام الأعمدة. هناك الدالة stem التي تستخدم كثيرا في مجال المعالجة الرقمية للإشارات. شكل (٥,٢٤) يبين استخدام هذه الدالة في رسم الخطوات التالية:

```
>> x=[0:0.5:30];
>> y=sin(x)./sqrt(x+1);
>> stem(x,y)
```



شكل (٥,٢٣). الرسم بالدالة contour.

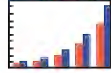


شكل (٥،٢٤). استخدام الدالة stem في الرسم.

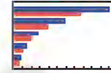
وهناك الكثير من هذه الدوال التي لا يتسع المكان لتجربتها كلها، ولكن سنكتفي بعرض القليل منها مع عرض ملخص لهذه الدوال كما في شكل (٥،٢٥). شكل (٥،٢٦) وشكل (٥،٢٧) يبينان الكثير من دوال الرسم بأنماط مختلفة فحاول تجربة كل منها ورؤية التأثيرات المختلفة لها نظرا لضيق المساحة لعرض تفاصيل كل هذه الدوال.

Bar Graphs

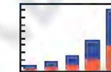
bar (grouped)



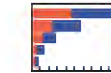
barh (grouped)



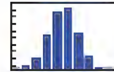
bar (stacked)



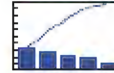
barh (stacked)



hist



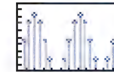
pareto



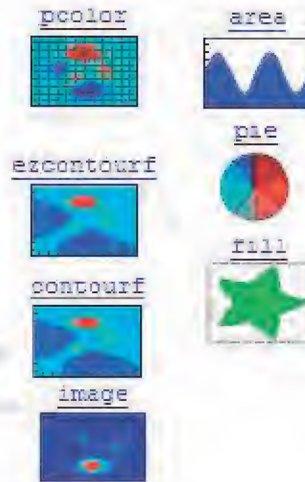
errorbar



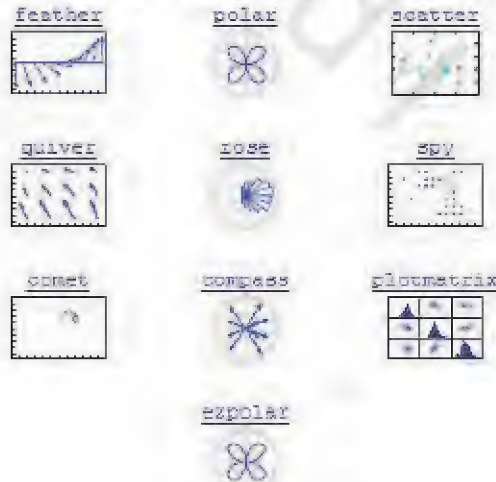
stem



شكل (٥،٢٥). دوال الرسم بالأعمدة.



شكل (٥,٢٦). دوال رسم المساحات.



شكل (٥,٢٧). دوال مختلفة لرسم أنماط مختلفة من الأشكال.

فيما يلي بعض التمارين التي توضح للقارئ استخدام الماتلاب في الرسم.

تمارين محلولة

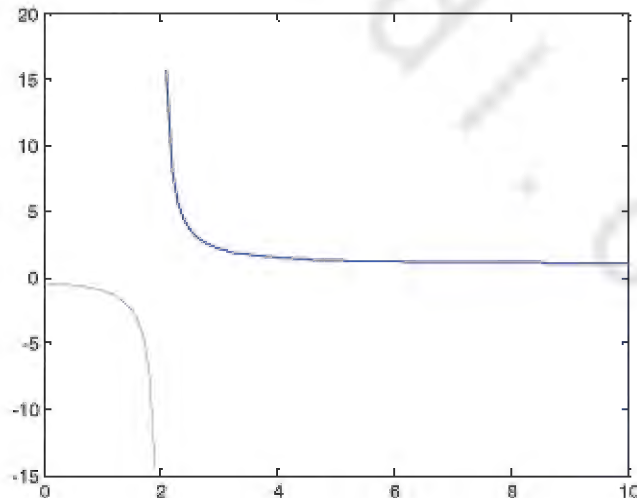
٥-١ اكتب برنامجا لرسم الدوال التالية :

١- $u = \frac{\sin x}{x} \quad for\ 0 \le x \le 10$

٢- $v = \frac{x^2 + 2}{x^2 - 4} \quad for\ 0 \le x \le 10$

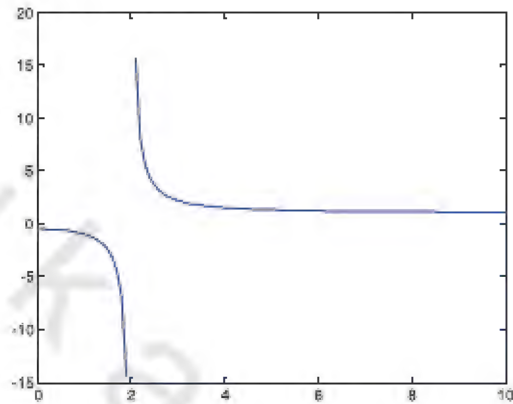
٣- $w = \frac{(5-x)^{1/2} + 2}{(4 - x^2)} \quad for\ 0 \le x \le 10$

```
%Solution of Excersis 5-1-1
% Draw te graph of the function u=sin(x)/x
x=0:0.1:10;
u=sin(x)./x;
figure, plot(x,u)
```



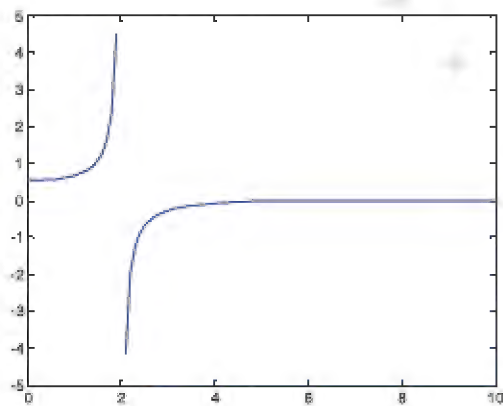
شكل (٥,٢٨). تمرين ٥-١-١.

```
%Solution of Excersis 5-1-2
% Draw te graph of the function v=(x^2+2)/(x^2-4)
x=0:0.1:10;
v=(x.^2+2)./(x.^2-4);
figure, plot(x,v)
```



شكل (٥,٢٩). تمرين ٥-١-٢.

```
%Solution of Excersis 5-1-3
% Draw te graph of the function w=(5-x).^(1/2)./(4-x.^2)
x=0:0.1:10;
w=(5-x).^(1/2)./(4-x.^2);
figure, plot(x,w)
```

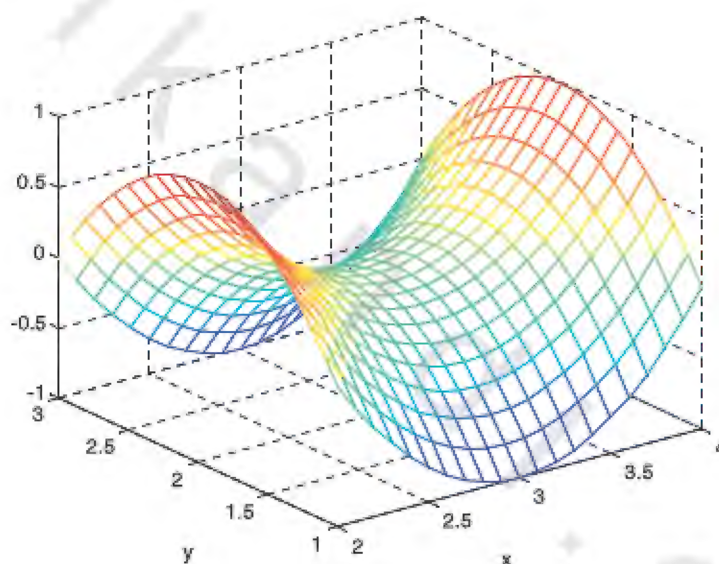


شكل (٥,٣٠). تمرين ٥-١-٣.

٥-٢ اكتب برنامجا لرسم السطح المعرف بالدالة الآتية:

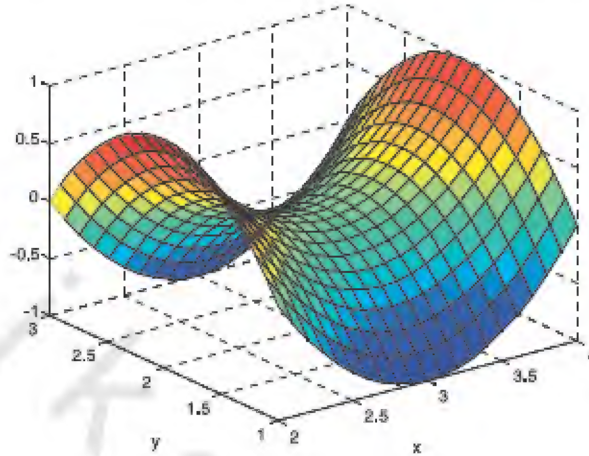
$$f(x,y) = (x-3)^2 - (y-2)^2 \quad for\ 2 \le x \le 4\ and\ 1 \le y \le 3$$

```
%Solution of Excersis 5-2
% Plot the surface defined by the function f(x,y)=(x-
3)^2-(y-2)^2
[X,Y]=meshgrid(2:0.1:4,1:0.1:3);
Z=(X-3).^2 - (Y-2).^2;
figure(1), mesh(X,Y,Z), xlabel('x'), ylabel('y')
```



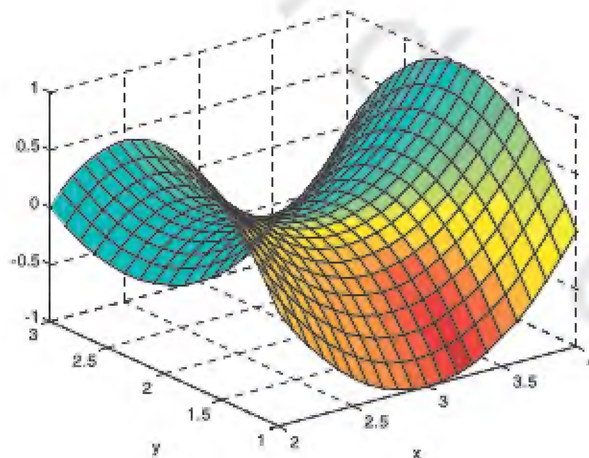
شكل (٥,٣١). تمرين ٥-٢.

```
figure(2), surf(X,Y,Z), xlabel('x'), ylabel('y')
```



شكل (٥,٣٢) تمرين ٥-٢.

```
figure(3), surfl(X,Y,Z),xlabel('x'), ylabel('y')
```

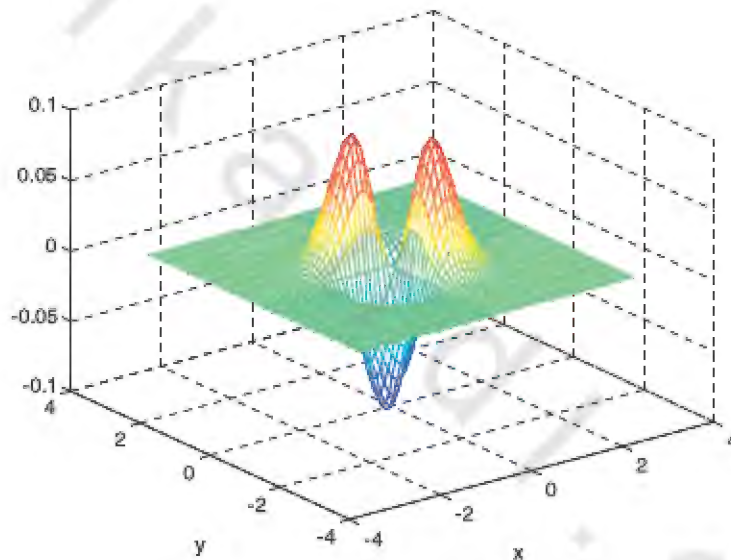


شكل (٥,٣٣). تمرين ٥-٢.

٥-٣ اكتب برنامجا لرسم السطح المعرف بالدالة الآتية :

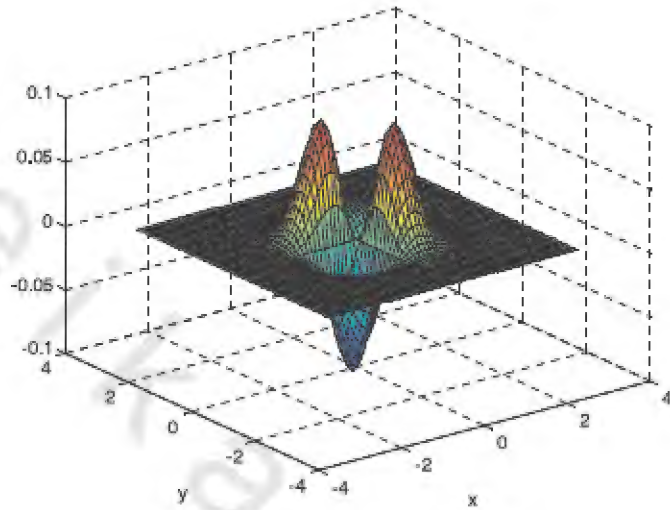
$$f = -xye^{-2(x^2+y^2)} \quad for \ -3 \le x \le 3 \ and \ -3 \le y \le 3$$

```
%Solution of Excersis 5-3
% Plot the surface defined by a function
[X,Y]=meshgrid(-3:0.1:3,-3:0.1:3);
f=-X.*Y.*exp(-2*(X.^2+Y.^2));
figure(1), mesh(X,Y,f),xlabel('x'), ylabel('y')
```



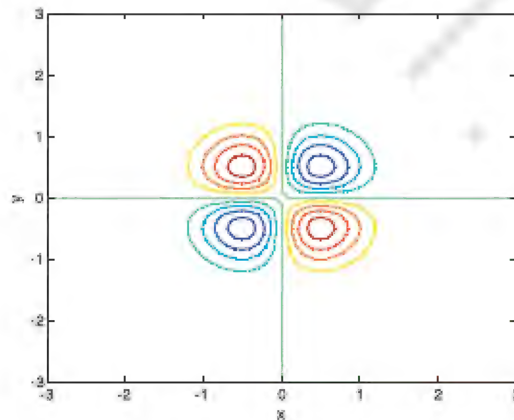
شكل (٥،٣٤). تمرين ٥-٣ باستخدام الدالة mesh.

```
figure(2), surf(X,Y,f),xlabel('x'), ylabel('y')
```



شكل (٥,٣٥). تمرين ٥–٣ باستخدام الدالة surf.

```
figure(3), contour(X,Y,f),xlabel('x'), ylabel('y')
```



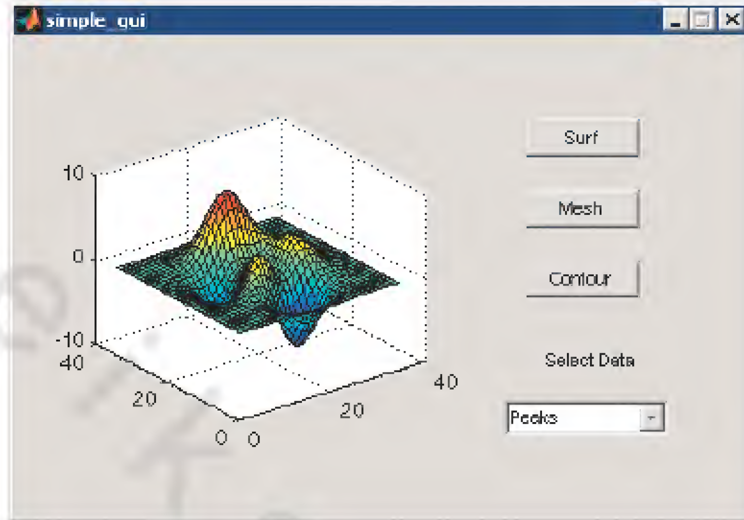
شكل (٥,٣٦). تمرين ٥–٣ باستخدام الدالة contour.

الفصل السادس

شاشات التعامل مع المستخدم

(٦,١) مقدمة

تحتوي شاشات التعامل مع المستخدم على العديد من المكونات التي تمكن المستخدم من التعامل تفاعليا مع البرامج من خلال طرق مختلفة، تتلخص في ضربة مفتاح، أو الاختيار من العديد من صناديق الاختيار، أو تحريك أحد الأزرار المنزلقة وغيرها الكثير. الميزة هنا أن المستخدم ليس من الضروري أن يكون على دراية بكتابة أمر معين من أوامر الماتلاب في نافذة الأوامر command window أو حتى يكون مضطرا لكتابة برنامج معين في ملف من ملفات الإم. إنه فقط سينقر على أحد المفاتيح أو يختار بالفأرة من أحد الاختيارات. إن أي شخص قد كتب برامجاً باستخدام لغة البايسيك المرئي أو لغة أخرى من لغات البرمجة المرئية سيقدر قيمة هذه الإمكانية التي يقدمها برنامج ماتلاب. في هذا الفصل نشرح كيفية استخدام محرر التصميم لبناء شاشة للتعامل مع المستخدم، ثم بعد ذلك – وبدون الدخول في الكثير من التفاصيل – نقوم ببناء نفس الشاشة من خلال مجموعة من الأوامر.



شكل (٦،١). مثال على إحدى شاشات التعامل مع المستخدم.

شكل (٦،١) يبين مثالا على أحد شاشات التعامل مع المستخدم. في هذا الشكل نلاحظ وجود مساحة لعرض أحد الأشكال الثلاثية الأبعاد، ويتم اختيار هذا الشكل من قائمة تسقط بمجرد النقر على السهم الموجود على يمينها. بعد اختيار الشكل يمكن اختيار طريقة رسمه من خلال النقر على أحد أزرار الضغط إما surf للعرض في صورة سطح، أو mesh للعرض في صورة شبكة، أو contour للعرض في صورة خارطة ارتفاعات. نؤكد هنا على أن المستخدم يحصل على ذلك دون كتابة أي أمر، أو كتابة أي برنامج، أو حتى دون أي دراية ببرنامج ماتلاب على الإطلاق، إنه يتعامل مع هذه الشاشة كما لو كان يتعامل مع أحد تطبيقات الميكروسوفت، مثل برنامج الوورد أو البوربوينت.

إن تصميم إحدى شاشات التعامل مع المستخدم لابد أن يمر بالخطوات التالية :

١- لابد أن تجيب عن السؤال التالي : ماذا يريد المستخدم أن يرى على هذه الشاشة، وما هي الاختيارات المتاحة أمامك كمبرمج؟

٢- بعد أن تتمكن من الإجابة عن هذا السؤال يمكنك أن تترجم ذلك إلى مكونات تقوم بتوزيعها على شاشة العرض المتاحة. بالطبع سيترك لك حرية توزيع هذه المكونات على الشاشة من حيث أماكنها والمسافات بينها؛ لأن ذلك سيختلف من مصمم لآخر تبعا لرغبة المصمم.

٣- بعد توزيع المكونات على الشاشة عليك أن تقوم بكتابة البرمجيات التي سيتم استدعاؤها للتنفيذ بمجرد النقر على أي مكون و اختياره.

هذه هي الثلاثة خطوات الأساسية التي لابد من المرور بها عند تصميم أي شاشة من شاشات التعامل مع المستخدم. مثل هذا النوع من البرمجة يسمى البرمجة المحددة بهدف، والتي يقصد بها كتابة البرنامج لأداء هدف معين ينفذ عند الطلب فقط. هناك أكثر من طريقة يمكن إتباعها في تصميم شاشات التعامل مع المستخدم، وسنرى هذه الطرق في الأجزاء التالية.

(٦,٢) بناء شاشة تعامل مع المستخدم باستخدام محرر تصميم شاشات التعامل

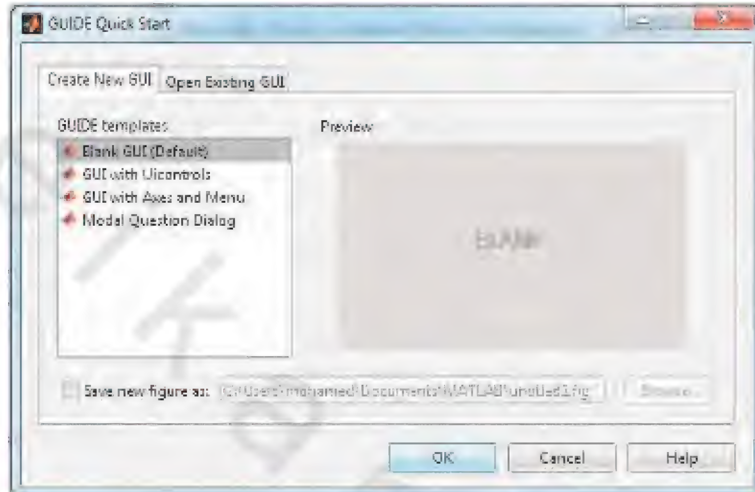
يتم بناء شاشات التعامل مع المستخدم بهذه الطريقة بإتباع الخطوات التالية:

١- في ساحة عمل الماتلاب اكتب الأمر guide حيث ستظهر أمامك الشاشة الموضحة في شكل (٦,٢). تحتوي هذه الشاشة على شريط عنوان يحتوي العنوان GUIDE Quick Start مع الثلاثة أزرار الخاصة بالتحكم، والتي توجد في أي شاشة تعمل تحت مظلة نوافذ الميكروسوفت. في أول خطوة سنختار الاختيار التلقائي Blank GUI (Default) عن طريق النقر على زر OK.

هناك طرق عديدة يمكن منها أن نبدأ برنامج GUIDE ومنها:

١- من مساحة العمل بكتابة GUIDE كأمر من أوامر ماتلاب.
٢- من قائمة الملفات File، نختار New ومنها نختار GUI.
٣- من قائمة Start ثم اختيار MATLAB ومنها نختار GUIDE(GUI Builder).
٤- من شريط الأيقونات نختار أيقونة GUIDE وننقر عليها.

بمجرد بدء برنامج GUIDE بأي من هذه الطرق ستظهر أمامك الشاشة الموضحة في شكل (٦،٢).



شكل (٦،٢) أول خطوة في إنشاء شاشة جديدة للتعامل مع الحاسب.

٢- بمجرد النقر على زر OK ستظهر لك شاشة تحرير كالموضحة في شكل (٦،٣) حيث تحتوي هذه الشاشة على مساحة تحرير، سيتم وضع جميع المكونات عليها كما سنرى في الخطوات التالية. على يسار هذه الشاشة يوجد صندوق لهذه المكونات التي سنختار منها ما نشاء ونضعه على شاشة التحرير. لاحظ أن هذه الشاشة أصبح لها اسم وهو untitled1.fig لحين تغيير هذا الاسم وتخزينه في المكان المناسب، وهذا الاسم موجود في شريط العنوان، والذي يليه شريطان آخران أحدهما شريط قوائم يحتوي القوائم الشهيرة في كل تطبيقات نوافذ ميكروسوفت، ثم شريط أيقونات يحتوي العديد من الأيقونات المعروفة بجانب بعض الأيقونات الخاصة التي سيتم شرحها فيما بعد مع بيان خطوات التصميم. الأدوات الموجودة في شريط أو صندوق الأدوات على يسار الشاشة يمكن التعرف على اسم كل منها بمجرد الوقوف عليه

بالفأرة. أو يمكن إظهار اسم كل منها بجوار الأيقونة الخاصة به عن طريق النقر على قائمة الملفات file ثم النقر على اختيار preference ستظهر لك شاشة جديدة انقر على خيار بيان أسماء المكونات show names in component palette ثم انقر OK سيظهر لك شريط المكونات بحيث يظهر كل مكون واسمه بجواره إذا كنت تفضل هذا المظهر. يمكن التحكم في مساحة شاشة التحرير بالوقوف بالفأرة على المربع الأسود في أسفل يمين الشاشة ثم سحب الفأرة حيث ستلاحظ تغيير مساحة الشاشة مع عملية السحب. كان ذلك يمثل الخطوة الأولى من تصميم شاشات التعامل مع المستخدم، وهي تجهيز شاشة العرض وضبط مساحتها.



شكل (٦،٣). إظهار شاشة التحرير وشريط الأدوات.

٣- الخطوة الثانية من خطوات تصميم شاشات التعامل مع المستخدم هي وضع المكونات على الشاشة التي تم ضبطها في الخطوتين السابقتين. من شريط المكونات سنسحب الزر الضاغط ثلاث مرات ونضعه في الأماكن التقريبية كما هو موضح في شكل (٦،٤).

٤- ابدأ بإضافة المكونات التالية إلى شاشة العرض :

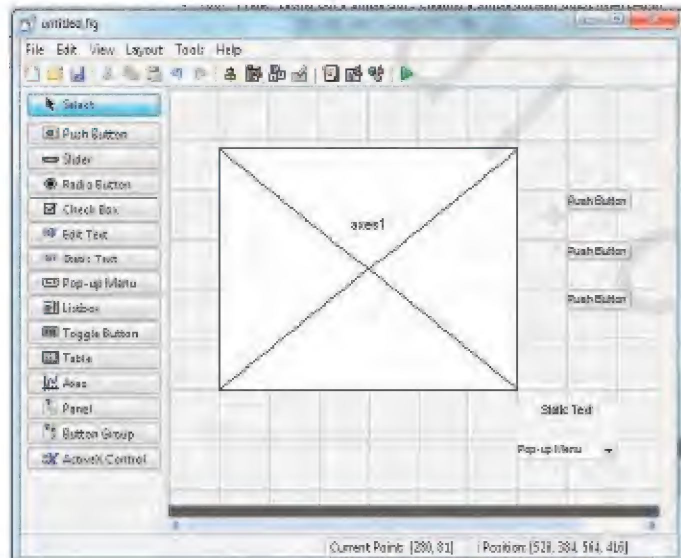
مساحة نص ثابتة Static text area

قائمة منسدلة pop up menu

محاور Axes

كل هذه المكونات موضحة على شكل (٦,٤)، لاحظ أننا في هذا الشكل قد غيرنا طريقة ظهور المكونات في شريط المكونات بحيث يكون اسم كل مكون بجانب هذا المكون.

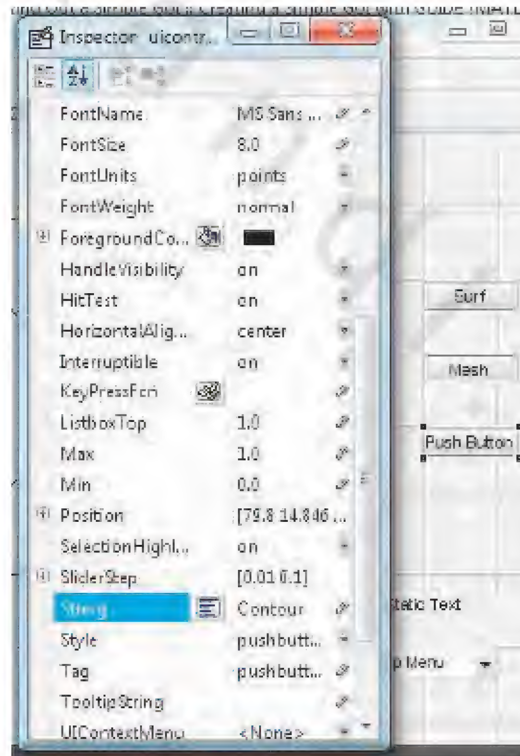
يمكنك محاذاة المكونات المختلفة عن طريق اختيارها جميعا (المكونات المراد محاذاتها) بالنقر على هذه المكونات مع الضغط على زر Ctrl، ثم النقر على قائمة الأدوات Tools ثم عرض قائمة المحاذاة بالنقر على display alignment tools. من هذه القائمة يمكن وضع المحاذاة أو اختيار نمط المحاذاة سواء المحاذاة الأفقية أو الرأسية، وكذلك يمكنك ضبط المسافات بين المكونات وبعضها سواء كانت هذه المسافات رأسية أو أفقية.



شكل (٦,٤) شاشة التعامل مع المستخدم بعد إضافة مكونات مختلفة إليها.

٥- بعد إضافة المكونات على الشاشة سنبدأ في تسمية كل مكون من هذه المكونات:

• مسمى الزر الأول يتم تغييره من Push Button إلى مسطح أو Surf والزر الثاني سنعطيه الاسم شبكة Mesh والثالث سيعطي الاسم Contour. يتم ذلك بالنقر على قائمة View من شريط القوائم ثم اختيار فاحص الخواص Property Inspector. بعد ذلك ننقر على الزر المراد تغيير اسمه وليكن الزر الأول، ثم نذهب إلى الخاصية String من فاحص الخواص ثم نقف فيها ونغيرها إلى الكلمة Surf. انقر في أي مكان خارج مجال الخاصية String ستجد أن اسم الزر قد تغير إلى الاسم الجديد. اضغط على الزر الثاني ثم الثالث وكرر نفس الخطوات السابقة لتغيير أسماء باقي الزراير كما في شكل (٦,٥).



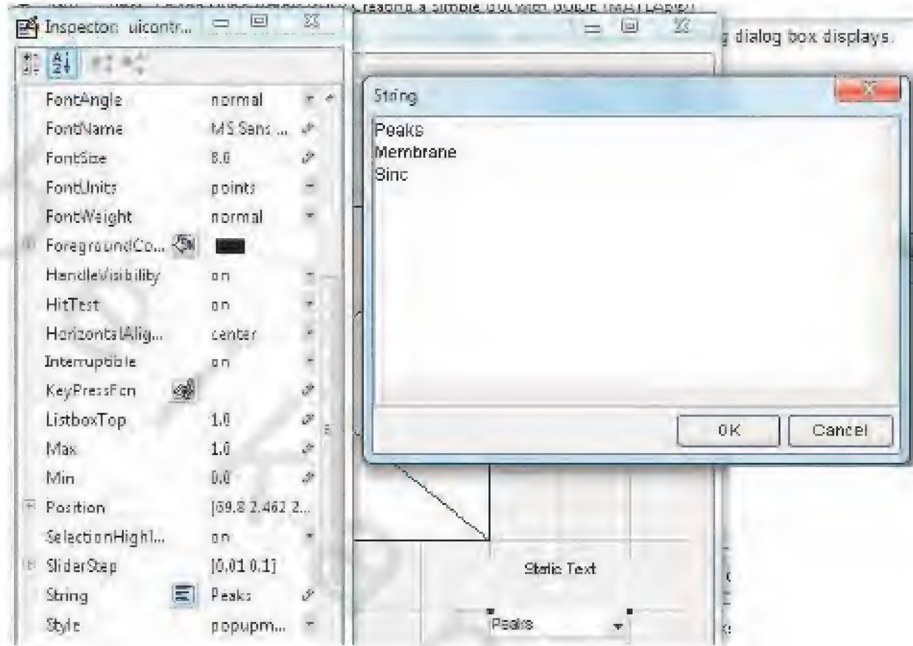
شكل (٦,٥). تغيير اسم المكونات من خلال شريط فاحص الخواص.

• مسمى القائمة ومحتوياتها يتم تغييره عن طريق النقر عليها ثم الذهاب إلى الشريط المقابل للخاصية String وبالنقر على هذه الخاصية سيفتح لك مربع جديد ابدأ في إضافة مكونات القائمة في هذا المربع كما هو موضح في شكل (٦,٦). لاحظ أن محتويات هذه القائمة هي الدوال التي سيتم عرضها في مساحة الرسم أو بين المحاور وهي Peaks و Membrane و Sinc. بعد الانتهاء من إدخال هذه المكونات اضغط الزر OK لتسجيل هذه المكونات الجديدة.

• تغيير مسمى النص الثابت static text سيتم بنفس الطريقة عن طريق النقر عليه ثم الذهاب إلى الخاصية String وتغيير مسمى النص الثابت إلى Select Data ثم انقر OK.

• المرحلة الأخيرة من مراحل إنشاء شاشة التعامل مع المستخدم – بعد الانتهاء من إدخال مسميات جميع المكونات – هي تخزين هذه الشاشة، ويتم ذلك من خلال النقر على زر التشغيل الأخضر المائل يمينا، حيث سيظهر لك مربع حواري يخبرك بتخزين هذه الشاشة. يتم التخزين في ملفين أحدهما يحتوي الشكل نفسه في ملف بامتداد .fig والآخر ملف إم بامتداد .m . بمجرد النقر على هذا الزر الأخضر سيظهر لك مربع حواري يخبرك أنه بهذا الاختيار سيتم تخزين الشكل في ملفين كما ذكرنا، وفي هذه الحالة عليك النقر على الاختيار yes لتستمر في عملية التخزين من خلال اختيار مسمى لهذا الشكل وقد اخترنا المسمى SimpleGUI. يمكنك أيضا اختيار مسار آخر غير المسار التلقائي الذي يختاره ماتلاب إذا أردت ذلك.

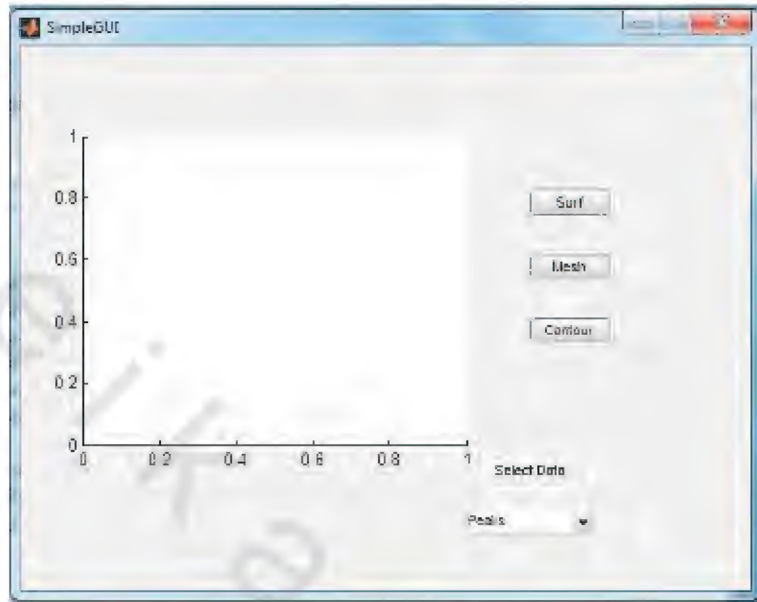
بالانتهاء من عملية التخزين ستظهر لك شاشة التعامل مع المستخدم في صورتها النهائية كما في شكل (٦,٧) حيث يمكنك تجربة جميع هذه الأزرار وقائمة العرض. ولكن تذكر أننا لم نبرمج هذه المكونات حتى الآن؛ لذلك فإنك سترى أنه بالنقر على أي زر لن يتغير أي شيء حتى تتم برمجة هذه المكونات في الخطوة التالية.



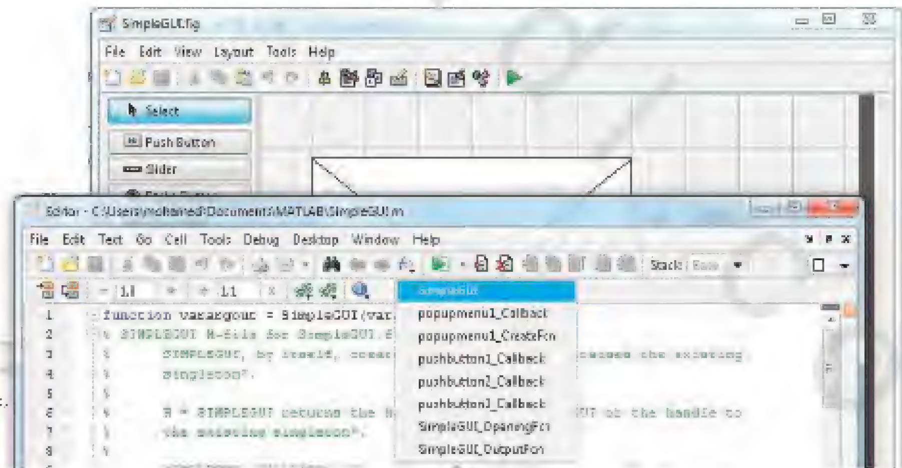
شكل (٦.٦). إدخال محتويات قائمة العرض.

٦- أول خطوات برمجة لهذه المكونات هي توليد البيانات المراد رسمها في مساحة العرض. سيتم توليد هذه البيانات في دالة تسمى دالة الافتتاح. دالة الافتتاح هذه تكون أول ما ينفذ من قبل ماتلاب في ملف الإم الخاص بأي شاشة تعامل مع المستخدم يتم توليدها باستخدام المرشد GUIDE الذي استخدمناه في إنشاء هذه الشاشة. سنرى الآن كيفية إضافة بيانات الثلاث دوال peaks, membrane, sinc في دالة الافتتاح.

٧- من المفروض أن هناك ملفاً اسمه SimpleGUI.m موجوداً من خطوة التخزين السابقة، إذا لم يكن هذا الملف مفتوحا فحاول فتحه من قائمة View ثم اختار M file Editor ثم انقر على الأيقونة f في شريط الأيقونات كما في شكل (٦.٨) حيث ستسقط لك قائمة تحتوي اسم الشاشة التي نعمل فيها، ومحتوياتها من أزرار تحكم وقائمة العرض ودالة الافتتاح.



شكل (٦,٧). الشكل النهائي لشاشة التعامل مع المستخدم.



شكل (٦,٨). برمجة مكونات شاشة التعامل مع المستخدم SimpleGUI.

٨- من هذه القائمة الساقطة انقر على الاختيار SimpleGUI_OpeningFcn حيث ستلاحظ وقوف دليل الكتابة cursor عند بداية دالة الافتتاح في البرنامج simpleGUI.m حيث ستجد الأوامر التالية :

```
% --- Executes just before SimpleGUI is made visible.
function SimpleGUI_OpeningFcn(hObject, eventdata,
handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future
version of MATLAB
% handles    structure with handles and user data (see
GUIDATA)
% varargin   command line arguments to SimpleGUI (see
VARARGIN)

% Choose default command line output for SimpleGUI
handles.output = hObject;

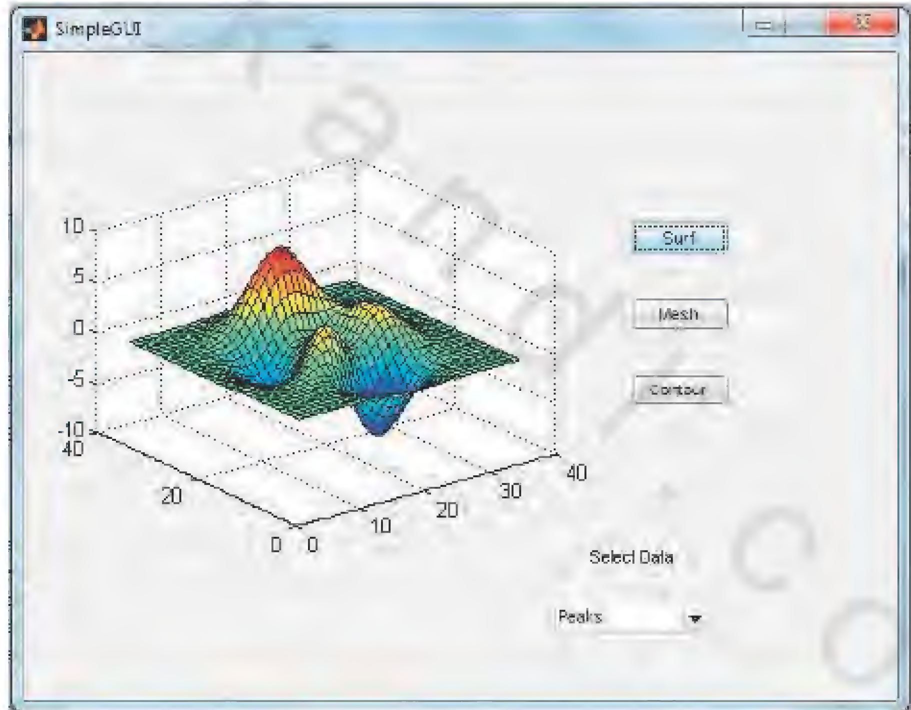
% Update handles structure
guidata(hObject, handles);
```

٩- أضف الأوامر التالية التي ستحسب بيانات المنحنيات المراد رسمها مباشرة بعد التعليق الذي يبدأ بكلمة ٪varagin.... في دالة الافتتاح كما يلي :

```
% varargin   command line arguments to SimpleGUI (see
VARARGIN)
% Create the data to plot.
handles.peaks=peaks(35);
handles.membrane=membrane;
[x,y] = meshgrid(-8:.5:8);
r = sqrt(x.^2+y.^2) + eps;
sinc = sin(r)./r;
handles.sinc = sinc;
% Set the current data value.
handles.current_data = handles.peaks;
surf(handles.current_data)
% Choose default command line output for SimpleGUI
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);
```

١٠-الستة خطوط الأولى من هذا الكود تولد البيانات للدوال peaks, membrane, sinc باستخدام الماتلاب. إنها تقوم بتخزين البيانات في منشأ التداول handles structure حيث سيتم تمريرها إلى جميع برمجيات تنفيذ المكونات، فمثلا يمكن لأزرار الضغط الثلاثة أن تنادى على هذه البيانات من منشأ التداول. آخر سطرين في هذا الكود تولد بيانات الدالة peaks وتعرض مسطح surf هذه الدالة كما في شكل (٦،٩) كرسم تلقائي عند تشغيل برنامج simpleGUI.



شكل (٦،٩). الشاشة التلقائية للتعامل مع المستخدم بعد برمجتها.

١١- في هذه الخطوة سنبرمج قائمة العرض بحيث تعرض الدالة المراد رسمها عند اختيارها من داخل هذه القائمة. عندما يقوم المستخدم باختيار واحد من الرسومات الثلاثة، فإن ماتلاب يضع الخاصية Value في شريط الخواص بحيث تساوي رقم الرسم المطلوب peaks, membrane, contour. ثم يجعل بيانات التداول الحالي handles.current_data بحيث تعرض هذا الرسم. لكي نصل إلى برمجية قائمة العرض إذا لم تكن مفتوحة أمامك من الخطوة السابقة، فإنه يمكنك عمل ذلك بالنقر بالزرار الأيمن للفأرة على قائمة العرض، حيث ستنزل أمامك قائمة أخرى خاصة بهذا المكون، اختر منها View Callback ومنها اختار Callback، حيث سيفتح أمامك برنامج الـ simpleGUI إذا لم يكن مفتوحا وسيقف دليل الكتابة عند برمجية قائمة العرض التي تحتوي الأسطر التالية:

```
% --- Executes on selection change in popupmenu1.
function popupmenu1_Callback(hObject, eventdata,
handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future
version of MATLAB
% handles    structure with handles and user data (see
GUIDATA)

% Hints: contents = get(hObject,'String') returns
popupmenu1 contents as cell array
%        contents{get(hObject,'Value')} returns selected
item from popupmenu1
```

١٢- أضف أسطر الكود التالية في برمجية قائمة العرض بعد السطر الذي يبدأ بـ %handles...:

```
% handles    structure with handles and user data (see
GUIDATA)
% Determine the selected data set.
str = get(hObject, 'String');
val = get(hObject,'Value');
% Set current data to the selected data set.
```

```
switch str{val};
case 'Peaks' % User selects peaks.
   handles.current_data = handles.peaks;
case 'Membrane' % User selects membrane.
   handles.current_data = handles.membrane;
case 'Sinc' % User selects sinc.
   handles.current_data = handles.sinc;
end
% Save the handles structure.
```

١٣- سنبرمج الأزرار الثلاثة بنفس طريقة برمجة قائمة العرض عن طريق استدعاء برمجية كل منها ثم إضافة الأكواد التالية إليها بعد السطر الذي يبدأ بكلمة handles٪ في كل منها كما يلي : بالنسبة للزر surf أضف السطرين التاليين :

```
% Display surf plot of the currently selected data.
surf(handles.current_data);
```

بالنسبة للزر Mesh أضف السطرين التاليين :

```
% Display mesh plot of the currently selected data.
mesh(handles.current_data);
```

بالنسبة للزر contour أضف السطرين التاليين :

```
% Display contour plot of the currently selected data.
contour(handles.current_data);
```

١٤- بذلك نكون قد انتهينا من برمجة هذه الشاشة للتعامل مع المستخدم، ويمكن تخزينها الآن والتجربة معها من خلال التنفيذ باستخدام الزر الأخضر الموجود في نفس الشاشة، أو إذا لم تكن هذه الشاشة مفتوحة فإنه يمكن التنفيذ من سطح المكتب أو مساحة التشغيل بكتابة اسم البرنامج SimpleGUI كما ننفذ أي برنامج من مساحة التشغيل.

بذلك نكون قد انتهينا من تقديم مثال متكامل على بناء شاشة تعامل مع المستخدم باستخدام طريقة المرشد GUIDE. هناك طرق أخرى لبرمجة شاشات التقابل مع المستخدم عن طريق البرمجيات ولكنها في الغالب لا تكون بنفس سهولة هذه الطريقة حيث إنها تولد الشاشة بالكامل من خلال أوامر تكتب في ملف إم. للحصول على النتيجة السابقة ننصح المستخدم بكتابة الدالة التالية وتخزينها ثم استدعائها بكتابة اسم الدالة من مساحة العمل :

```
function simple_gui
   %  Create and then hide the GUI as it is being
constructed.
   f =
figure('Visible','off','Position',[360,500,450,285]);
   %  Construct the components.
   hsurf =
uicontrol('Style','pushbutton','String','Surf',...
          'Position',[315,220,70,25],...
          'Callback',{@surfbutton_Callback});
   hmesh =
uicontrol('Style','pushbutton','String','Mesh',...
          'Position',[315,180,70,25],...
          'Callback',{@meshbutton_Callback});
   hcontour = uicontrol('Style','pushbutton',...
          'String','Countour',...
          'Position',[315,135,70,25],...
          'Callback',{@contourbutton_Callback});
   htext = uicontrol('Style','text','String','Select
Data',...
          'Position',[325,90,60,15]);
   hpopup = uicontrol('Style','popupmenu',...
          'String',{'Peaks','Membrane','Sinc'},...
          'Position',[300,50,100,25],...
          'Callback',{@popup_menu_Callback});
   ha =
axes('Units','Pixels','Position',[50,60,200,185]);

align([hsurf,hmesh,hcontour,htext,hpopup],'Center','None
');

   % Create the data to plot.
   peaks_data = peaks(35);
   membrane_data = membrane;
   [x,y] = meshgrid(-8:.5:8);
   r = sqrt(x.^2+y.^2) + eps;
   sinc_data = sin(r)./r;

   % Initialize the GUI.
   % Change units to normalized so components resize
automatically.
   set([f,ha,hsurf,hmesh,hcontour,htext,hpopup],...
   'Units','normalized');
   %Create a plot in the axes.
```

```
   current_data = peaks_data;
   surf(current_data);
   % Assign the GUI a name to appear in the window
title.
   set(f,'Name','Simple GUI')
   % Move the GUI to the center of the screen.
   movegui(f,'center')
   % Make the GUI visible.
   set(f,'Visible','on');
   %  Callbacks for simple_gui.
   function popup_menu_Callback(source,eventdata)
         % Determine the selected data set.
         str = get(source, 'String');
         val = get(source,'Value');
         % Set current data to the selected data set.
         switch str{val};
         case 'Peaks' % User selects Peaks.
            current_data = peaks_data;
         case 'Membrane' % User selects Membrane.
            current_data = membrane_data;
         case 'Sinc' % User selects Sinc.
            current_data = sinc_data;
         end
      end
   % Push button callbacks. Each callback plots
current_data in
   % the specified plot type.
   function surfbutton_Callback(source,eventdata)
   % Display surf plot of the currently selected data.
      surf(current_data);
   end
   function meshbutton_Callback(source,eventdata)
   % Display mesh plot of the currently selected data.
      mesh(current_data);
   end
   function contourbutton_Callback(source,eventdata)
   % Display contour plot of the currently selected
data.
      contour(current_data);
   end
end
```

الطريقة GUIDE تعتبر طريقة، سهلة حيث إن جميع المكونات تكون موجودة في صندوق أو في شريط، ويمكن سحب أي هذه المكونات ووضعها على الشاشة في المكان المناسب وبالحجم المناسب، حيث يمكن تغيير كل ذلك عن طريق السحب بالفأرة، وليس من خلال أوامر كما في الطرق الأخرى. نكتفي بهذا الكم عن إنشاء شاشات التفاعل مع المستخدم، وهناك الكثير من الإضافات التي يمكن الاستفادة منها لمن يريد من وسط المساعدة الخاص بالماتلاب في هذا الموضوع أو بالبحث في المراجع [٩-١١].

الفصل السابع

برنامج المحاكاة سميولينك

(٧,١) مقدمة

برنامج المحاكاة سميولينك simulink هو أحد منتجات نفس الشركة المنتجة للماتلاب، وهذا البرنامج يمكن استخدامه في محاكاة أي نظام أو مشكلة قد تخطر على بالك، بدءا من السؤال الذي قد يخطر على بالك عن هذه المشكلة إلى وضع نموذج لحل هذه المشكلة يتم فيه محاكاة الدخل والخرج لهذا النظام إلى متابعة سلوك النظام من حيث علاقة الخرج بالدخل مع الزمن. كل ذلك يتم من خلال بلوكات وظيفية مختلفة يتم سحبها بالفأرة ووضعها في نموذج متكامل. يدعم هذا البرنامج الكثير من التطبيقات الميكانيكية والكهربية ومعالجة الإشارات والصور بما في ذلك التطبيقات الخطية منها وغير الخطية والممثلة في الزمن المستمر أو في الزمن المتقطع. الكثير من العلماء والمهندسين في الكثير من التطبيقات يبدأ عادة في محاكاة مشكلته على هذا البرنامج ودراستها جيدا، ورؤية استجابة هذا النظام عند أماكن مختلفة من خلال أجهزة عرض يتم وضعها عند الأماكن المراد دراستها، وذلك قبل أن يبدأ في التنفيذ العملي لها. يحتوي ماتلاب على العديد من البرامج التجريبية المتكاملة التي يمكنك النداء عليها من نافذة الأوامر command window من خلال المساعدة simulink demo models. يبدأ هذا الفصل بشرح كيفية تشغيل برنامج

السميولينك مع مثال لعمل نموذج بسيط، ثم نعطى مثالا تطبيقيا على العمليات المنطقية، وبعد ذلك نتعرض للأنظمة الفرعية. محاكاة المعادلات الحسابية ودوال العبور للأنظمة من التطبيقات المهمة، والتي نتناولها بالشرح أيضا. ينتهي هذا الفصل بتناول كيفية تنشيط الأنظمة و إضافة طرف قدح للأنظمة الفرعية.

أمثلة لما يمكن أن يتم محاكاته باستخدام برنامج السميولينك:

- محاكاة السفن ودراسة أدائها في الظروف المختلفة.
- محاكاة المركبات الجوية ديناميكيا ودراسة أدائها في الظروف المختلفة.
- محاكاة السيارات ودراسة أدائها أيضا في الظروف المختلفة.
- محاكاة أسواق المال وتغيراتها اليومية.
- محاكاة أنظمة الاتصالات المختلفة.
- محاكاة الأنظمة الحيوية الطبية.

عملية المحاكاة أو النمذجة لأي نظام أو لأي مشكلة في وسط السيميولينك تمر بالمراحل التالية:

١- تحديد النظام أو المشكلة المراد محاكاتها.

٢- تحديد مكونات هذا النظام.

٣- محاولة نمذجة هذا النظام بمعادلة.

٤- استخدام السميولينك في بناء الرسم الصندوقي للنظام.

٥- تنفيذ هذا النموذج.

٦- التحقق من نتائج هذا النظام.

لاحظ أن أول ثلاث خطوات يتم تنفيذها خارج نطاق السميولينك وقبل أن تبدأ في استخدام هذا البرنامج.

في الخطوة الأولى يتم دراسة النظام المقترح، وهل هذا النظام كبير بحيث يتم تجزئته إلى أنظمة أصغر، ويتم بناء كل واحد من هذه الأنظمة الصغيرة، ثم يتم تجميعها كلها في نظام واحد يتم تجربته في النهاية.

في خطوة تحديد مكونات النظام، هناك ثلاثة أنواع من المكونات:

- المعاملات، أو الثوابت التي تظل ثابتة دائما إلا إذا قمت أنت بتغييرها.
- المتغيرات، أو القيم المتغيرة التي تتغير مع الزمن.
- الإشارات، وهي إشارات الدخل والخرج بين بلوكات السميولينك والتي تتغير ديناميكيا مع الزمن أثناء المحاكاة.

لكل واحد من هذه الأنظمة الأساسية أو الجانبية عليك أن تسأل نفسك وتجيب عن هذه الأسئلة:

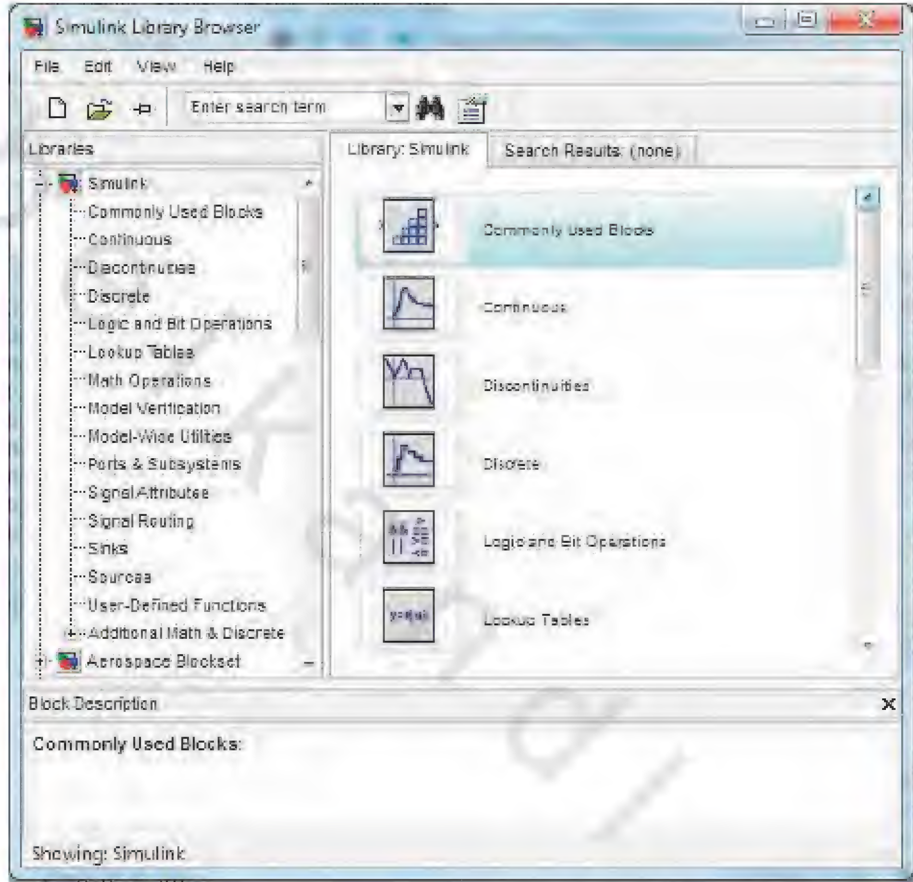
- كم عدد إشارات الدخل لهذا النظام؟
- كم عدد إشارات الخرج لهذا النظام؟
- كم عدد المتغيرات في هذا النظام؟
- كم عدد الثوابت في هذا النظام؟
- هل هناك إشارات مرحلية أو داخلية في هذا النظام؟

بمجرد الإجابة عن هذه الأسئلة تكون مستعدا الآن للبدء في نمذجة النظام.

(٧,٢) بدء تشغيل السميولينك

برنامج سميولينك هو برنامج متكامل مع برنامج الماتلاب، ولذلك لكي تبدأ برنامج السميولينك عليك أولا تشغيل برنامج ماتلاب ثم من نافذة الأوامر اكتب الأمر simulink حيث ستظهر الشاشة الموضحة في شكل (٧,١).

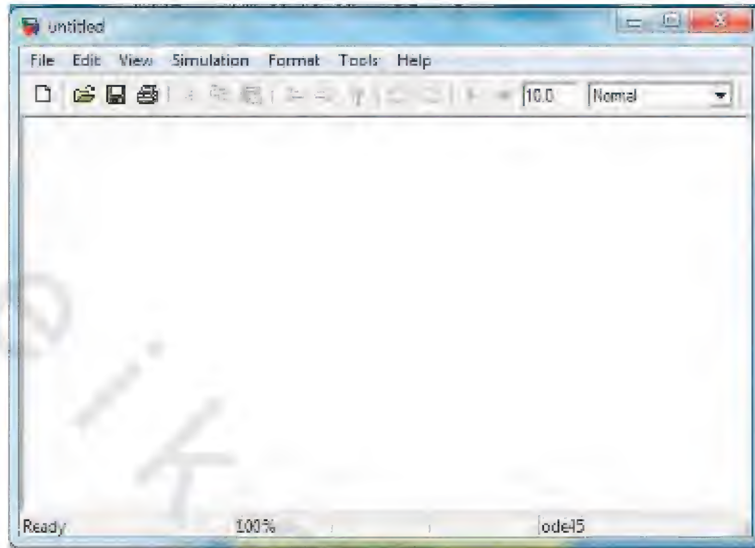
```
>> simulink
```



شكل (٧,١). أول شاشة تظهر مع بدء تشغيل سميولينك.

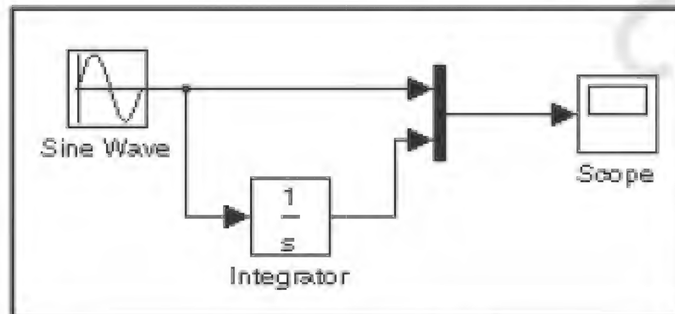
كما يمكنك تشغيل السميولينك بالنقر على الأيقونة الخاصة به في شريط الأيقونات في شاشة ماتلاب الرئيسة ، أو من قائمة start ثم الذهاب إلى simulink.

الآن يمكن أن نبدأ في عمل موديل جديد من قائمة الملفات file في شاشة مكتبة سميولينك اختر new ثم اختر model ستظهر لك مساحة عمل جديدة كالموضحة في شكل (٧,٢).



شكل (٧,٢). مساحة العمل لنموذج جديد.

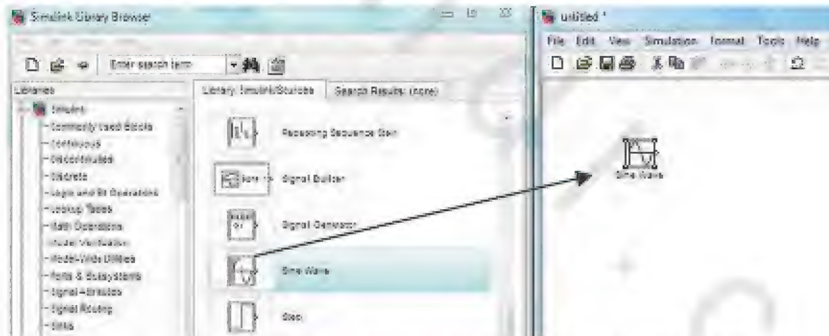
كمثال افتتاحي نفهم منه كيفية النمذجة داخل برنامج سميولينك وخطواتها ؛ سنبدأ بالمثال التالي البسيط الموضح في شكل (٧,٣) والذي يتكون من مصدر مولد ذبذبات جيبية ، حيث سيتم تكامل خرج هذا المصدر وجمعه وعرضه على مبين ذبذبات scope. خطوات تنفيذ هذا النموذج البسيط ستكون كالتالي :



شكل (٧,٣) مثال افتتاحي بسيط.

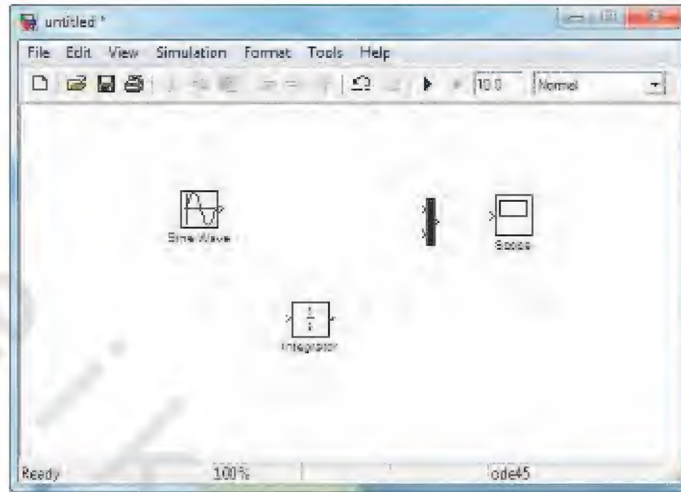
١- على مساحة العمل للنموذج الجديد، والمسمى untitled إلى أن نغير اسمه فيما بعد سنحتاج لإضافة المكونات التالية:

- مصدر موجة جيبية من مكتبة المصادر sources. بالنقر مرتين على أيقونة المصادر الموجودة في مستكشف المكتبة كما في شكل (٧,١) ستفتح لك مكتبة أخرى من المصادر العديدة كالموضحة في شكل (٧,٤)، اختر منها المصدر Sine Wave واسحبه بالفأرة إلى النموذج واتركه في المكان المناسب كما في شكل (٧,٤).
- مبين ذبذبات scope من مكتبة البالوعات Sinks سنسحب أيقونة الـ Scope ونضعها في مكان مناسب على النموذج بنفس الطريقة السابقة.
- بلوك تكامل Integrator من مكتبة الزمن المستمر Continuous.
- بلوك توزيع Mux من مكتبة توزيع الإشارات Signal Routing.



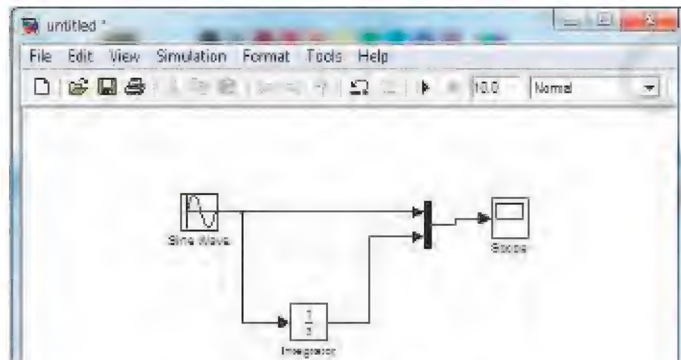
شكل (٧,٤). سحب أيقونة مصدر الموجة الجيبية إلى النموذج.

٢- بعد إضافة كل هذه المكونات - كما في شكل (٧,٥) - يمكن إعادة توزيعها على النموذج عن طريق السحب بالفأرة إلى أن تصل إلى الوضع المرضى بالنسبة لك.



شكل (٧,٥). وضع كل المكونات على النموذج.

٣- بعد وضع المكونات على النموذج نقوم بالتوصيل فيما بينها ويتم ذلك عن طريق الوقوف بالفأرة على نقطة التوصيل في البلوك الأول، ثم السحب بالفأرة حتى تصل إلى نقطة التوصيل في البلوك الثاني. أو بالنقر على البلوك الأول لاختياره، ثم مع الضغط على الزر Ctrl يتم النقر على البلوك الثاني، فيقوم ماتلاب بالتوصيل فيما بينهما مباشرة. شكل (٧,٦) يبين نفس النموذج بعد إتمام عملية التوصيل.



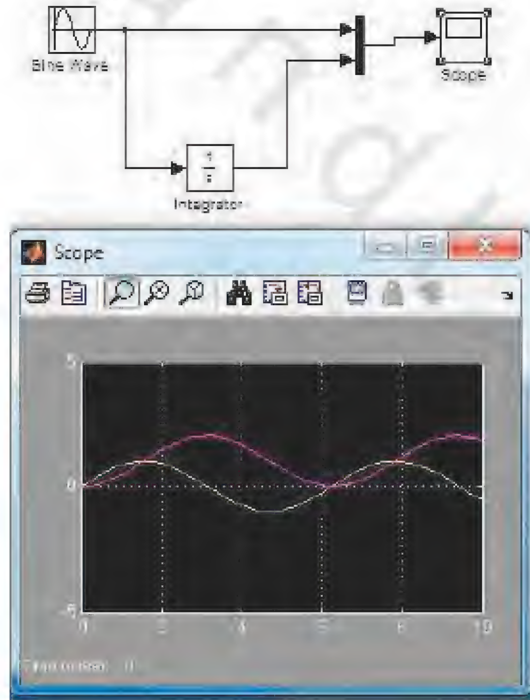
شكل (٧,٦). نفس النموذج بعد إتمام التوصيل.

٤- عن طريق النقر مرتين على أي واحد من هذه المكونات ستفتح لك نافذة يمكن من خلالها ضبط معاملات هذا المكون أو هذا البلوك. فمثلا يمكن ضبط مقدار وتردد وطور مصدر الإشارة الجيبية بالنقر مرتين عليه.

٥- الآن يمكن تخزين هذا النموذج بالنقر على قائمة الملفات file ثم Save as حيث يمكنك تخزين النموذج بأي مسمى تريده، ولقد أسميناه ex1.

٦- قبل تنفيذ النموذج انقر على قائمة Simulation ومنها اختر Configuration parameters حيث سيفتح لك مربع حوار يمكن من خلاله ضبط بعض معاملات التنفيذ مثل زمن بدء التنفيذ، وزمن نهاية التنفيذ وغير ذلك.

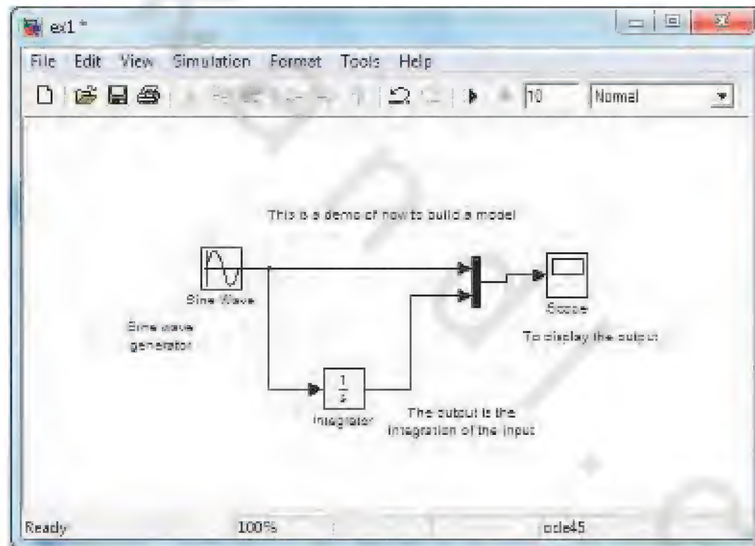
٧- بعد ضبط هذه المعاملات يمكن تنفيذ النموذج بالنقر على زر التشغيل (السهم الأسود المائل يمينا) في شاشة النموذج.



شكل (٧.٧). تنفيذ النموذج وعرض الخرج.

٨- بالنقر مرتين على أيقونة الـ scope سيظهر الخرج الذي من المفروض أن يظهر كما في شكل (٧،٧).

٩- في النهاية، يمكنك إضافة نصوص توضيحية على النموذج في أي مكان تريد عن طريق الوقوف بالفأرة على المساحة المراد الكتابة عندها والنقر مرتين، حيث ستظهر لك مساحة لإدخال النص يمكنك البدء في كتابة النص المطلوب. تذكر أنه لسوء الحظ، فإن ماتلاب ليس به إمكانية كتابة مثل هذه النصوص باللغة العربية. شكل (٧،٨) يبين نفس النموذج السابق بعد إضافة بعض النصوص التوضيحية عليه.

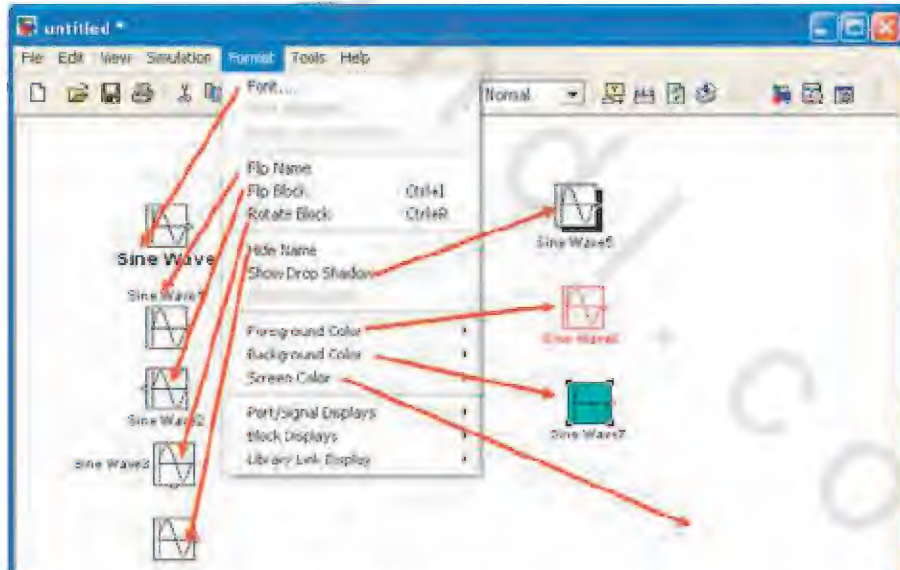


شكل (٧،٨). النموذج السابق بعد إضافة بعض التعليقات النصية.

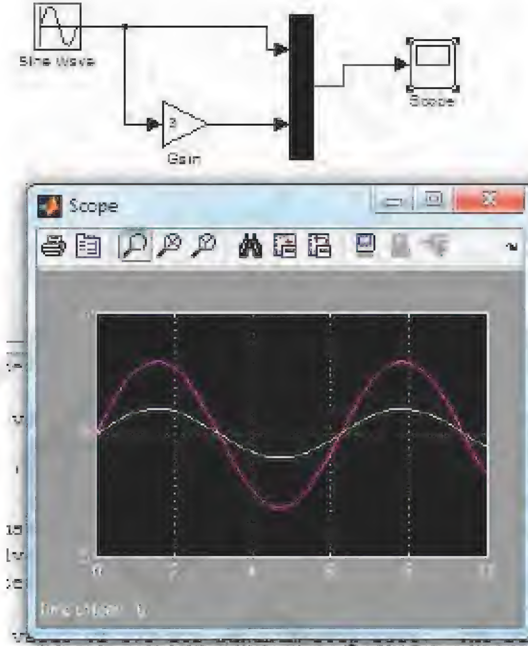
١٠- من قائمة التشكيل format في شاشة النموذج يمكنك تغيير لون البلوكات وتغيير خلفية النموذج كله كما يمكنك تدوير البلوك أو قلبه. شكل (٧،٩) يبين الكثير من التشكيلات التي يمكن إجراؤها على أي بلوك.

١٢- يمكنك ضبط المساحة التي يشغلها أي بلوك عن طريق النقر على هذا البلوك ثم من على النقاط التي تميز اختيار هذا البلوك يمكنك الوقوف عليها بالفأرة لزيادة طول أو عرض البلوك إلى الحجم الذي تريده.

١٣- يمكنك إدخال أي بلوك في أي مكان في النموذج عن طريق سحب هذا البلوك وإسقاطه على خط التوصيل الذي تريد وضعه عنده. فمثلا في شكل (٧,١٠) قمنا بإدخال المكبر Gain على الخط الواصل بين مصدر الإشارة والدخل الثاني لبلوك التوزيع بحيث بمجرد إسقاطه في هذه المساحة قام ماتلاب بتوصيله كما نرى في الشكل. انقر مرتين على المكبر واضبط معامل تكبيره على القيمة ٣ ونفذ البرنامج ثم انقر على الـ scope مرتين لترى الخرج كما في شكل (٧,١٠).



شكل (٧,٩). الكثير من عمليات التشكيل يمكن إجراؤها على أي بلوك.

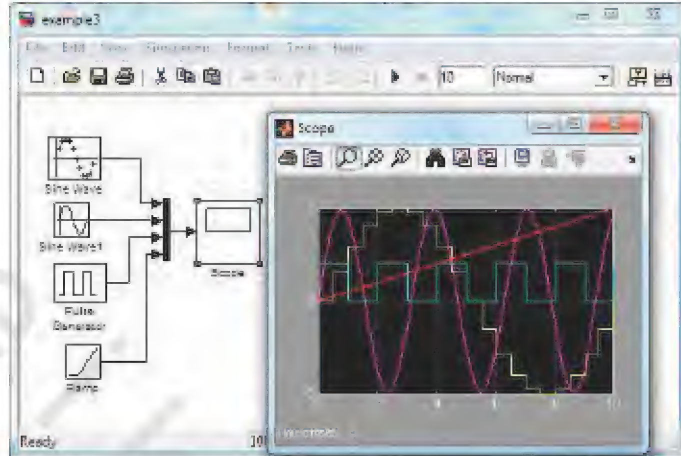


شكل (٧،١٠) التحكم في حجم البلوك وإدخال بلوك على النموذج.

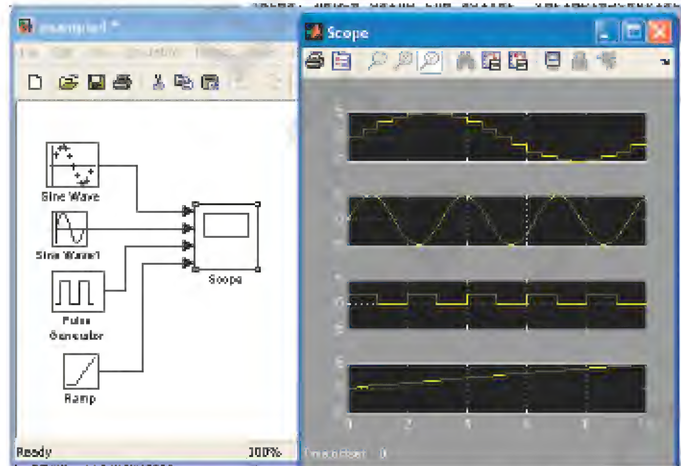
١٤- يمكنك إزالة أي بلوك أو خط توصيل بين اثنين بلوك عن طريق النقر على هذا المكون لاختياره ثم باستخدام المقص، أو الزر Del أو الزر Backspace يمكنك إزالة هذا المكون من على النموذج.

١٥- يمكنك إضافة أي بيانات نصية على أي خط توصيل عن طريق النقر مرتين على هذا الخط حيث سيفتح لك مربع نص يمكنك أن تكتب فيه ما تريد.

١٦- من البلوكات الشائعة الاستخدام مبين الذبذبات أو scope الذي يقوم بعرض إشارة الدخل مع الزمن. يمكن للأوسولوسكوب أن يكون له محور زمني واحد يعرض إشارة زمنية أو أكثر كما رأينا في الأشكال السابقة. كما يمكن عرض أكثر من إشارة بحيث يكون لكل إشارة محور زمني خاص بها. شكل (٧،١١) يبين أربع إشارات مجموعة على موزع إشارات ومعروضة على الأوسولوسكوب بمحور زمني واحد.



شكل (٧.١١). عرض أربع إشارات على محور زمني واحد.

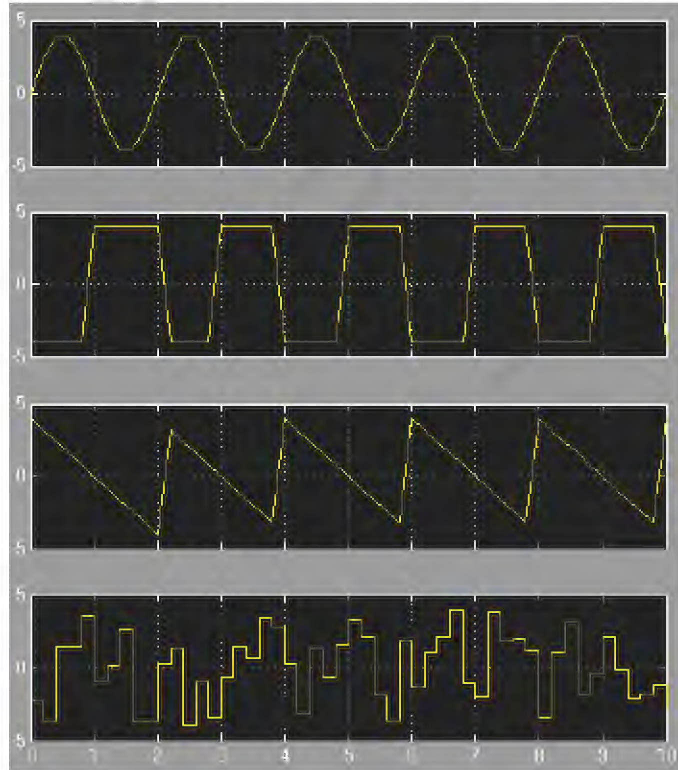


شكل (٧.١٢) عرض أكثر من إشارة بحيث تكون كل إشارة على محور زمني منفصل.

١٧- شكل (٧.١٢) يعرض نفس الأربع إشارات في شكل (٧.١١) ولكن هذه المرة يتم عرض كل إشارة على محور زمني واحد. نلاحظ من هذا الشكل أنه تم إدخال الإشارات مباشرة على الأوسولوسكوب مع تغيير عدد المحاور number of Y axes من شاشة الخواص إلى ٤ بدلا من واحد.

١٨- تحتوي شاشة عرض الأوسولوسكوب على الكثير من الأيقونات التي تستحق أن تجربها وترى تأثيرها. فهناك مثلا أيقونات التقريب لكل واحد من المحاور، وأيقونة عرض خواص الأسولوسكوب وغيرها فحاول استكشافها.

١٩- من البلوكات الكثيرة الاستخدام أيضا بلوك مولد الإشارات Signal Generator الذي يمكن الحصول منه على موجات جيبية أو موجات مربعة أو سن المنشار أو موجة عشوائية عن طريق الدخول في قائمة خواصه واختيار شكل الموجة المراد إخراجها. شكل (٧.١٣) يبين أربعة من هذه البلوكات، وقد تم ضبط كل منها على أحد هذه الموجات وعرضها جميعا على أوسولوسكوب واحد.



شكل (٧.١٣). موجات الخرج المختلفة من بلوك مولد الإشارات.

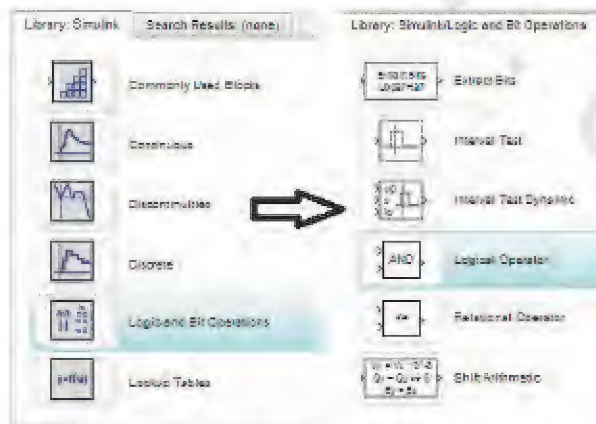
٢٠- يمكنك تغيير مدى المحاور الناتجة من الأوسولوسكوب ووضع عنوان له وذلك بالوقوف بالماوس على المحور الرأسي والنقر بالزرار الأيمن، حيث سيظهر لك مربع يمكنك من خلاله تغير القيم العظمى والصغرى للمحور ووضع عنوان لهذا الشكل.

٢١- بلوك التوزيع MUX له أكثر من دخل، وله خرج واحد، حيث يقوم الموزع باختيار هذه الإشارات الداخلة وإخراجها على خرج واحد الواحدة بعد الأخرى بحيث يمكن عرض أكثر من إشارة على الأوسولوسكوب. بالنقر المزدوج على هذا البلوك تفتح لك قائمة خواص يمكنك من خلالها اختيار عدد المداخل لهذا البلوك.

(٧,٣) العمليات المنطقية

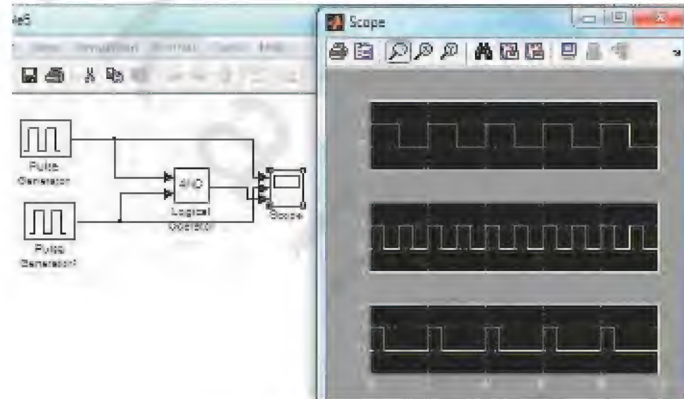
العمليات المنطقية والدوائر والأنظمة الرقمية من التطبيقات الكثيرة الاستخدام في العديد من التخصصات الهندسية وغير الهندسية. لذلك سنبدأ هنا في استخدام البلوكات المنطقية ونسوق من خلالها بعض الإمكانيات الأخرى للسميولينك.

من قائمة مكتبة سميولينك اختر أيقونة Logic and Bit Operations سيفتح لك قائمة كبيرة من العمليات المنطقية العديدة سنختار منها أيقونة Logical Operator كما في شكل (٧,١٤).

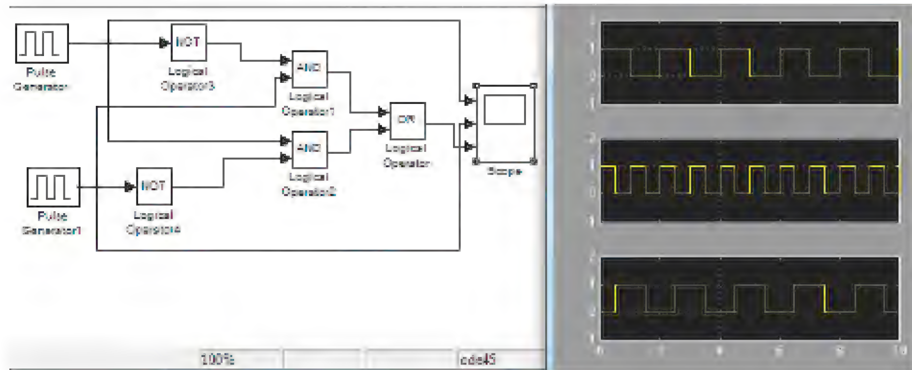


شكل (٧,١٤). من قائمة مكتبة سميولينك اختر أيقونة العمليات المنطقية.

بالنقر المزدوج على بلوك Logical Operator ستفتح قائمة خواص يمكنك من خلالها اختيار نوع العملية المنطقية AND, OR, NAND, NOR وغيرها من العمليات المنطقية الأساسية. اختر واحدة من هذه العمليات ولتكن عملية الـ AND واضبط عدد مداخلها على ٢ ، ثم نقوم بتوصيل مصدرين للنبضات على هذه الـ AND، ونوصل خرج هذه البوابة على أسولوسكوب لنرى نتيجة هذه العملية بالنسبة للدخل كما في شكل (٧،١٥). شكل (٧،١٦) يبين محاكاة البوابة XOR باستخدام بوابات الـ AND والـ NOT والـ OR مع عرض الخرج على الأوسولوسكوب. حاول بناء هذا النموذج وتجربته.



شكل (٧،١٥) محاكاة عملية الـــ AND.



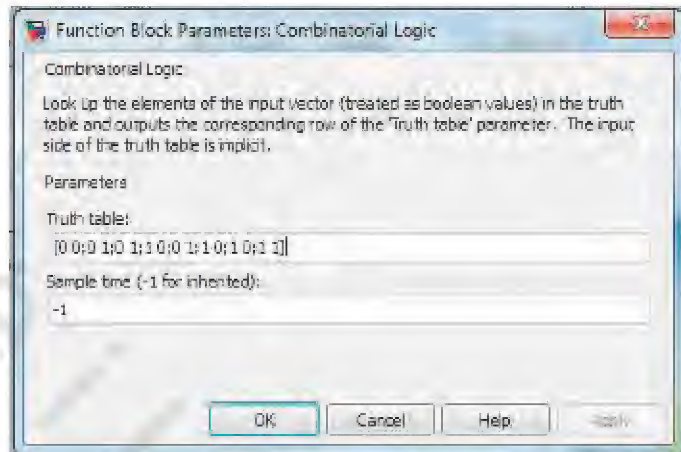
شكل (٧،١٦) محاكاة بوابة الـــ XOR.

بلوك المنطق التوافقي Combinational Logic يقوم بمحاكاة أي جدول حقيقة ينتج من أي عملية منطقية أو معادلة منطقية. في هذا البلوك يقوم المستخدم بتحديد مخارج جدول الحقيقة كمعاملات للبلوك في مقابل جميع الاحتمالات الممكنة للدخل بحيث يمثل كل دخل في صف. عدد صفوف جدول الحقيقة يساوي ٢ مرفوعة لأس يساوي عدد مداخل الدائرة المنطقية التي يمثلها جدول الحقيقة.

كمثال على ذلك، سنفترض جدول الحقيقة للمجمع الكامل الذي له ثلاثة مداخل a, b, Cn-1 وله خرجان أحدهما S الذي يمثل مجموع هذه الثلاث بتات و Cn الذي يمثل الحمل الناتج من هذا المجموع إن كان هناك حمل. جدول (٧،١) يبين تفاصيل هذا الجدول. لاحظ الترتيب المنطقي للدخل بدءا من الرقم ٠٠٠ حتى الرقم ١١١ وفي مقابل كل دخل تم وضع الدخل في العمودين S, Cn. شكل (٧،١٨) يبين محاكاة لدائرة منطقية مكونة من جدول حقيقة له ثلاثة مداخل وخرجان تم إظهارهما على أحد المظهرات الرقمية. لإدخال جدول الحقيقة ننقر مرتين على البلوك Combinational logic وفي المكان truth table ندخل صفى الخرج S, Cn-1 فقط في صورة صفوف مصفوفة يفصل كل منها فصلة منقوطة وموضوعة بين القوسين التاليين [ ] كما في شكل (٧،١٧) .

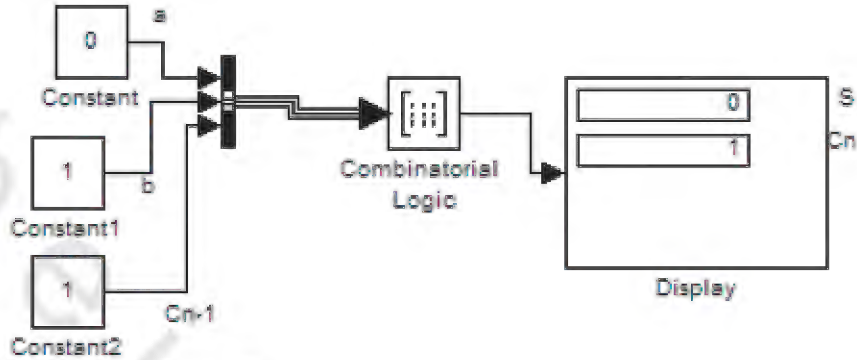
جدول (٧،١). جدول الحقيقة للمجمع الكامل.

a	b	Cn-1	S	Cn
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



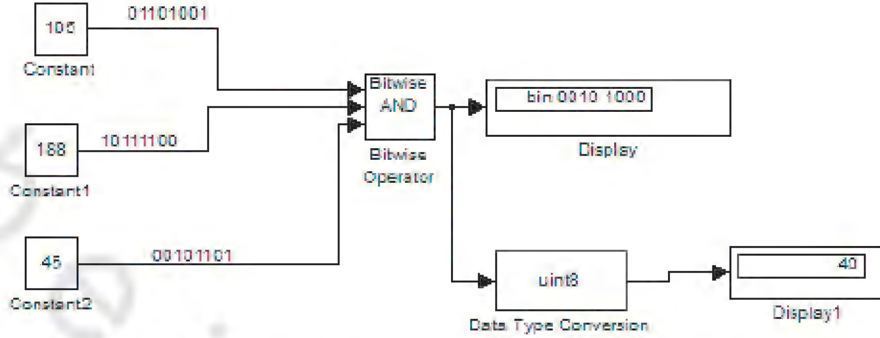
شكل (٧,١٧). إدخال جدول الحقيقة للبلوك التوافقي.

بلوك العرض display يعرض القيمة العددية للإشارة المدخلة على دخله، على العكس من الأوسولوسكوب الذي يعرض شكل الإشارة المدخلة مع الزمن. بالنقر المزدوج على هذا البلوك يمكن التحكم في عدد مداخله، وقد قمنا بضبط هذا العدد على ٢ مدخل التي تمثل مخارج المجمع الكامل. البلوك Bus creator في شكل (٧,١٨) يقوم بتجميع إشارات الدخل (في هذه الحالة ٣ مداخل) ويخرجها في صورة مسار متواز مجمع في خط واحد كما في الشكل حتى لا يزدحم الشكل بالخطوط. لاحظ أن عدد خطوط الخرج يساوي عدد المداخل، ويمكن التحكم في هذا العدد بالنقر المزدوج على البلوك وضبط هذا العدد. آخر بلوك في شكل (٧,١٨) هو البلوك Constant الذي يعطي قيمة ثابتة على الخرج يمكن تحديدها من شاشة مواصفات هذا البلوك التي تظهر عند النقر عليه مرتين. في هذه الشاشة انقر على شباك خواص الإشارة Signal attributes واختر نوع البيانات التي ينتمي إليها الثابت الذي ستخرج قيمته على خرج البلوك، فهذه البيانات يمكن أن تكون متضاعفة الدقة Double أو Single أو Fixed وهناك الكثير من أنواع البيانات التي يمكن الاختيار منها، ولكن في حالة التعامل مع البيانات الرقمية كما في هذا المثال فعليك أن تختار النوع Boolean.



شكل (٧.١٨). استخدام البلوك التوافقي لمحاكاة أي دائرة توافقية.

هناك البلوك Bitwise Operator الذي يقوم بإجراء العملية المنطقية التي تختارها (AND أو OR أو NAND أو ....) على البيانات المدخلة على مستوى البت ولا يعامل البيانات المدخلة على أنها رقم واحد كما رأينا في حالة البلوك Logical Operator. في شكل (٧.١٩) قمنا بإجراء عملية الـ AND على ثلاثة أرقام خارجة من ثلاثة بلوكات Constant ممثلة رقميا. أخذنا خرج الـ AND لعرضه على display في صورته الرقمية ، ثم من خلال محول بيانات قمنا بتحويل البيانات الرقمية الخارجة من الـ AND إلى الصورة العشرية وعرضها على شاشة العرض الثانية كما في الشكل. المصادر الثابتة Constant في شكل (٧.١٩) يجب ضبط بياناتها على النوع uint8 أي بيانات من ٨ بت بدون الإشارة بمعنى أن قيمة أي رقم من هذا النوع ستتراوح بين الصفر و ٢٥٥. يمكنك التجربة مع باقي العمليات الممكنة للبلوك Bitwise Operator.



شكل (٧,١٩). العمليات المنطقية على مستوى البت **Bitwise Operator**.

(٧,٤) الأنظمة الفرعية

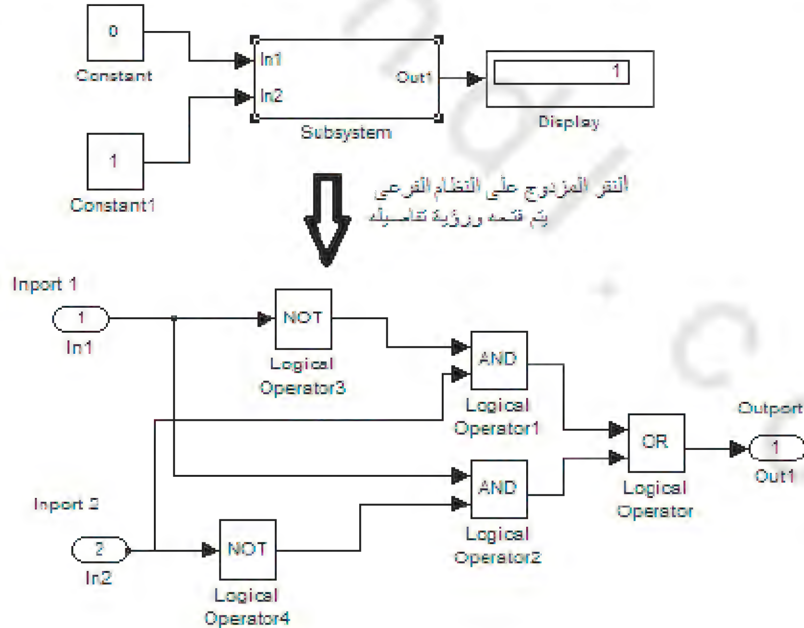
من الممكن أن يبلغ النموذج الذي تقوم ببنائه درجة عالية من التعقيد بحيث يكون من الصعب جدا تتبع تفاصيل هذا النظام في ظل افتراضه كنظام متكامل. في هذه الحالة يكون من المفيد جدا تجميع عدد من بلوكات النظام في نظام فرعي بحيث يمكن اعتبار هذا النظام الفرعي كما لو كان بلوك واحد، ومع تكرار ذلك يؤول النظام الأساسي إلى مجموعة من الأنظمة الفرعية. من مميزات هذه الطريقة أنها تسمح بتجميع البلوكات المتشابهة وظيفيا مع بعضها مما يسهل التعامل معها. كما أن هذه الطريقة تمكنك من بناء النظام الأساسي في صورة مجموعة من الأنظمة الطبقية أي الموضوعة في صورة طبقات يمكن التنقل فيها من طبقة لأخرى. يمكنك بناء النظام الفرعي بطريقتين:

الأولى: عن طريق إضافة بلوك نظام فرعي على النموذج المفتوح، ثم يتم فتح هذا البلوك وإضافة البلوكات الأساسية إليه.

الثانية: عن طريق إضافة البوكات الأساسية التي ستستخدم داخل النظام الفرعي على النموذج، ثم يتم تجميع هذه البلوكات في نظام فرعي.

١- توليد النظام الفرعي باستخدام بلوك النظام الفرعي

- من مكتبة الأطراف والأنظمة الفرعية Ports & Subsystems خذ نسخة من بلوك النظام الفرعي Subsystem وضعها على النموذج المفتوح.
- افتح النظام الفرعي Subsystem عن طريق النقر المزدوج عليه. في الحال سيفتح لك سميولينك نافذة نموذج جديدة.
- في نافذة النموذج الجديدة التي تم فتحها ابدأ في بناء النظام الفرعي.
- استخدم بلوكات الدخل Inport لتمثيل مداخل النظام الفرعي ، وبلوكات الخرج Outport لتمثيل مخارج النظام الفرعي.
- احفظ النظام الفرعي بالاسم الذي تريد كما في شكل (٧,٢٠)، حيث تم تخزين هذا النظام الفرعي باسم SubXOR حيث قمنا بوضع هذه البوابة في نظام فرعي.



شكل (٧,٢٠). بناء بوابة XOR داخل نظام فرعي.

• يمكنك الآن الخروج من شاشة النظام الفرعي المفتوحة والرجوع إلى النظام الفرعي نفسه، حيث بالنقر المزدوج عليه سيتم فتح تفاصيل هذا النظام لترى بوابة ال XOR التي قمت ببنائها من قبل.

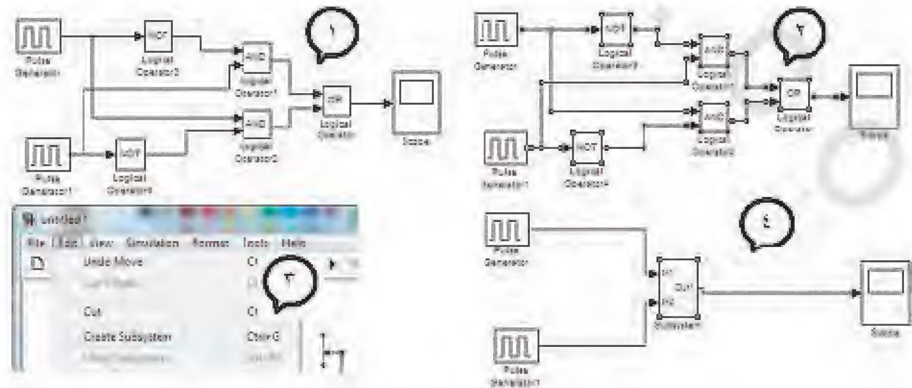
• يمكنك اختبار هذا النظام الفرعي عن طريق إضافة شاشة عرض على خرجه ومصدرين للدخل الثابت بعد ضبط بياناتهما على Boolean ثم وضع وحايد وأصفار على هذه المداخل للتحقق من جدول الحقيقة للبوابة XOR كما في شكل (٧,٢٠).

٢- توليد النظام الفرعي عن طريق تجميع البلوكات

في هذه الطريقة إذا كان لديك مجموعة من البلوكات التي تم تشغيلها واختبارها كنظام منفصل يمكنك تجميع هذه البلوكات في نظام فرعي كما يلي:

• جمع كل البلوكات والخطوط الموصلة بينها في داخل صندوق كبير عن طريق الوقوف بالفأرة خارج النظام، ثم النقر مع السحب لتغطية جميع البلوكات والخطوط الموصلة بحيث يتم اختيار جميع أجزاء النظام المراد وضعه في نظام فرعي. يمكنك عمل ذلك عن طريق اختيار مكونات النظام الواحد بعد الآخر، أو النقر على الأمر Select All.

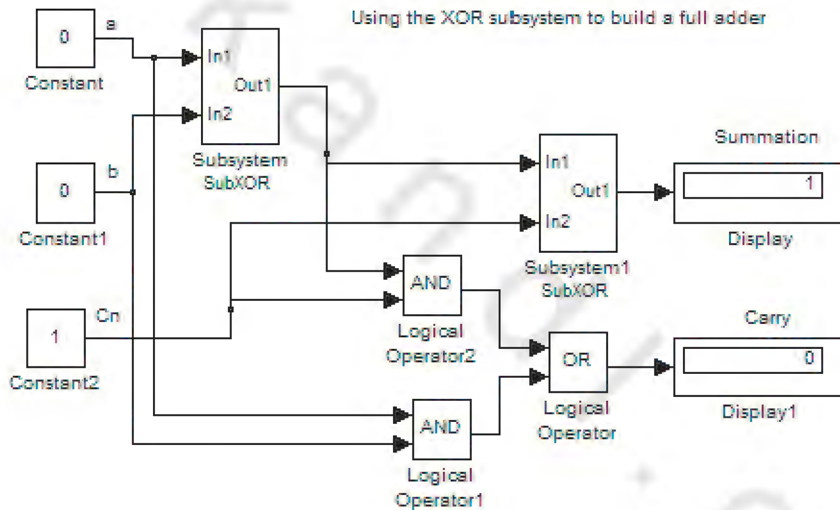
• من قائمة Edit انقر على الاختيار Create Subsystem حيث سيقوم سميولينك بتجميع كل هذه المكونات في نظام فرعي. شكل (٧,٢١) يبين خطوات هذه الطريقة.



شكل (٧,٢١). خطوات الحصول على نظام فرعي بالطريقة الثانية.

هذا النظام الفرعي الجديد سواء تم الحصول عليه بالطريقة الأولى أو الثانية يمكنك أن تعطيه أي اسم تريد عن طريق الوقوف عليه بالفأرة والنقر على الزر الأيمن حيث ستظهر لك قائمة اختر منها Block Properties حيث ستفتح لك نافذة حوار اختر منها Block Annotation ثم اختر الخاصية name وفي المقابل ضع الاسم الذي تريد، حيث سيظهر تحت أيقونة النظام الفرعي.

الآن سنستخدم هذا النظام الفرعي XOR مع بعض المكونات الأخرى لبناء دائرة مجمع كما في شكل (٧,٢٢).



شكل (٧,٢٢). دائرة مجمع كامل باستخدام بلوك النظام الفرعي XOR.

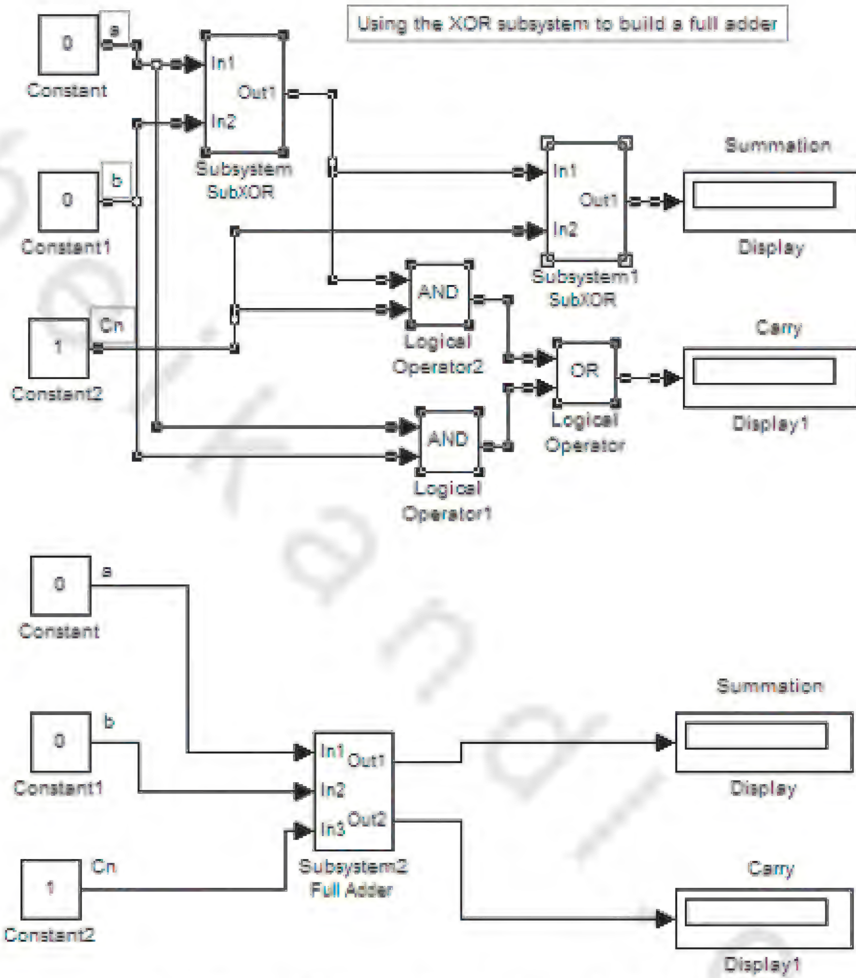
الآن سنختار كل المكونات الموجودة في شكل (٧,٢٢) ونجمعها داخل بلوك نظام فرعي جديد سنسميه Full Adder وهذا البلوك له ثلاثة مداخل وخرجان كما في شكل (٧,٢٣).

حتى الآن رأينا كيف تم تجميع بعض البلوكات لتكوين نظام فرعي يمثل بوابة XOR، بعد ذلك استخدمنا هذا النظام الفرعي مع بعض البلوكات الأخرى لبناء مجمع كامل، ثم وضعنا كل هذه المكونات في نظام فرعي جديد يمثل المجمع الكامل.

لكي نجمع الرقمين $A=a_3a_2a_1a_0$ و $B=b_3b_2b_1b_0$ سنحتاج لأربعة مجمعات كاملة Full Adder، المجمع الأول سيجمع البت a_0 مع البت b_0 وسنضع الدخل الثالث في هذا المجمع بصفر، خرج هذا المجمع سيكون S_0 و C_0. المجمع الثاني سيجمع a_1 مع b_1 مع الحمل من المرحلة السابقة C_0 وستعطي الخرج S_1 و C_1. المجمع الثالث سيجمع a_2 مع b_2 مع C_1 وسيعطي الخرج S_2 و C_2. أخيرا، المجمع الرابع سيجمع a_3 مع b_3 مع C_2 وسيعطي S_3 والحمل C_3. بنفس الطريقة يمكن زيادة وحدات تجميع أخرى لجمع أي بتات أخرى يراد جمعها.

شكل (٧,٢٤) يبين دائرة تجميع الرقمين A و B باستخدام بلوك النظام الفرعي Full Adder كوحدة بناء.

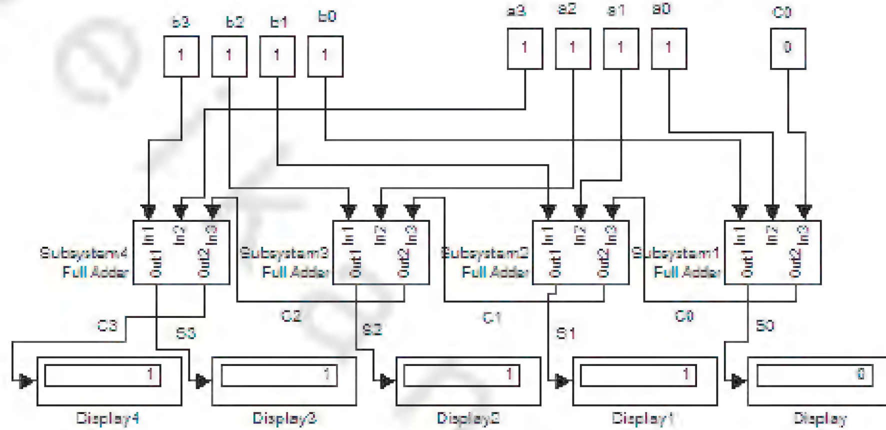
الآن، يمكن تجميع كل محتويات الدائرة الموجودة في شكل (٧,٢٤) لتصبح داخل بلوك نظام فرعي سنسميه مجمع ٤ بت، حيث يقوم هذا المجمع بجمع الرقمين A و B كل منهما ٤ بت ويعطي الناتج S المكون من ٤ بت أيضا والحمل C إلى المرحلة التالية كما في شكل (٧,٢٥). لاحظ في شكل (٧,٢٥) أن سميولينك قد أعاد ترتيب الأطراف بحيث تكون مرتبة داخل البلوك نفسه بحيث يكون الدخل الأول In1 من ناحية اليسار وبعده الدخل الثاني In2 وهكذا حتى الدخل التاسع In9 في أقصى اليمين. نفس الشيء تم بالنسبة للمخارج Out1 وبعده المخرج Out2 وهكذا حتى المخرج الأخير Out5 في أقصى اليمين. هنا أيضا نذكر أنه يمكنك تغيير مسميات الدخل والخرج إلى أي أسماء تريدها، فمثلا يمكن تسمية الدخل In1 بالاسم b0 والدخل In2 بالاسم a0 والدخل In3 بالاسم c0 وهكذا وذلك من خلال خواص هذا البلوك.



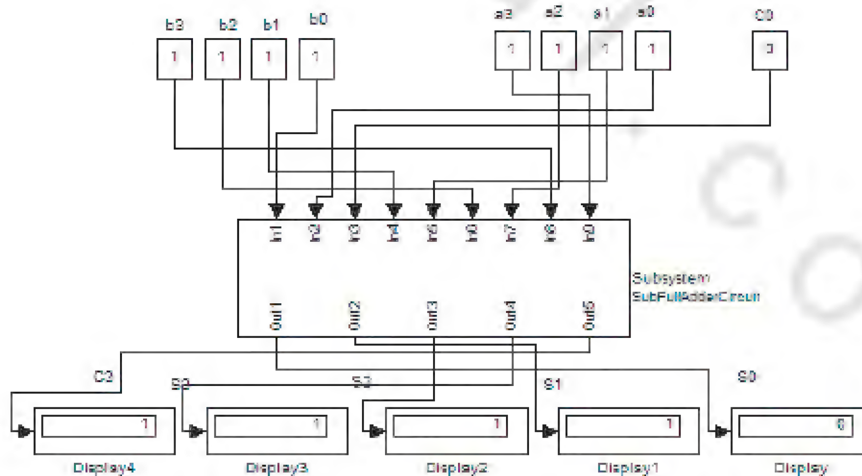
شكل (٧,٢٣). تجميع بلوك نظام فرعي للمجمع الكامل **Full Adder**.

لاحظ أنه بالنقر المزدوج على أي نظام فرعي فإنه يفتح شاشة تبين محتويات هذا النظام الفرعي، فمثلا بالنقر المزدوج على النظام الفرعي للمجمع ٤ بت SubFullAdderCircuit المبين في شكل (٧,٢٥) سيفتح لك شاشة تبين محتويات هذا البلوك الموضحة في شكل (٧,٢٤). وبالنقر المزدوج على أحد الأنظمة الفرعية الموجودة

في شكل (٧,٢٤) وهو FullAdder سيفتح لك شاشة جديدة تبين محتوياته كما في شكل (٧,٢٣) الذي يحتوي على أنظمة فرعية أخرى، وهي SubXor حيث بالنقر عليه سيفتح لك شاشة جديدة تبين محتويات هذا النظام الفرعي من بوابات الـ AND والـ OR. حاول تجربة ذلك.



شكل (٧,٢٤). بناء مجمع ٤بت باستخدام ٤ وحدات من بلوك النظام الفرعي **Full Adder**.



شكل (٧,٢٥). المجمع ٤ بت داخل نظام فرعي.

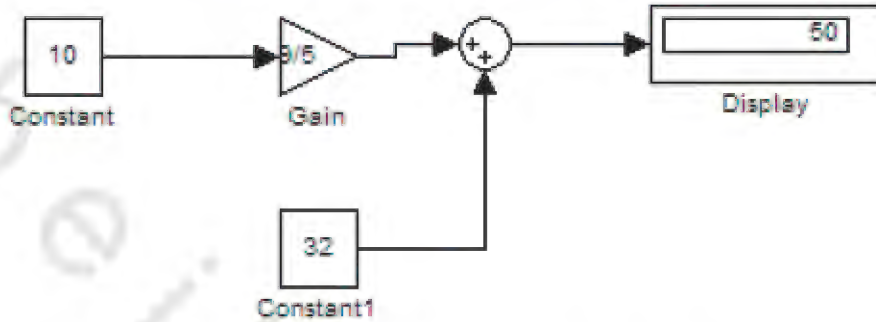
(٧,٥) محاكاة المعادلات الحسابية

من التطبيقات الكثيرة الاستخدام التي نحتاجها في الكثير من المجالات العلمية والهندسية بالذات هي التعامل مع المعادلات الحسابية ومحاكاتها، ونحن سنعرض هنا بعض هذه التطبيقات وبلوكات السميولينك المستخدمة في ذلك.

من قائمة مكتبة سميولينك انقر على بلوك العمليات الحسابية Math Operations حيث سيفتح لك قائمة كبيرة من بلوكات العمليات الحسابية. سنبدأ الحديث عن هذه البلوكات من خلال مثال لتحويل درجة الحرارة من التدريج المئوي إلى التدريج الفهرنهيتي تبعا للمعادلة التالية:

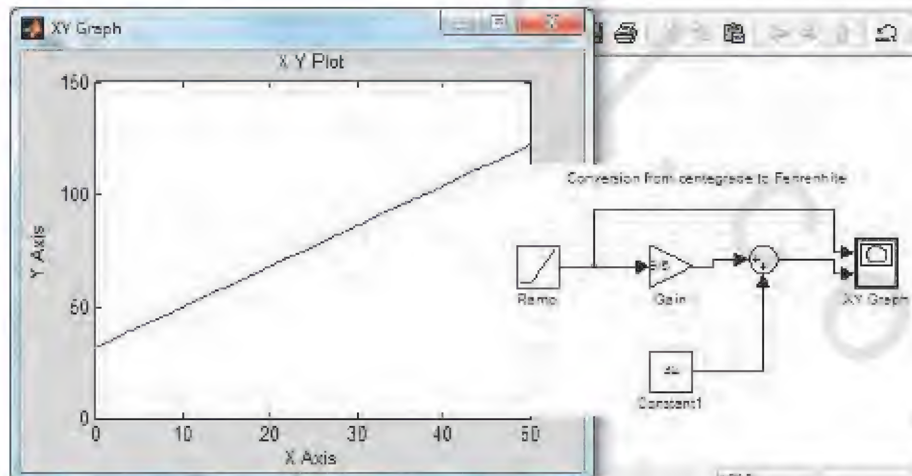
$$T_f = \frac{9}{5}T_c + 32$$

شكل (٧,٢٦) يبين طريقة لمحاكاة هذه المعادلة حيث تم استخدام مصدر الكمية الثابتة Constant من مكتبة المصادر Sources وبلوك المكبر Gain من مكتبة العمليات الحسابية Math operations والذي سيتم ضربه في الثابت الممثل لدرجة الحرارة المئوية، الكمية الثابتة ٣٢ تم استخدامها من المصدر الثاني Constant2. أخيرا بلوك التجميع Sum من بلوك العمليات الحسابية لتجميع هذين المسارين. النتيجة سيتم عرضها على عارض Display من بلوك النهايات Sinks في مكتبة سميولينك. الآن عليك وضع درجة الحرارة التي تمثل الدخل ولتكن ١٠ درجات مئوية كما في شكل (٧,٢٦) حيث ستكون النتيجة تساوي ٥٠ درجة فهرنهيتي كما في الشكل.



شكل (٧,٢٦). محاكاة معادلة التحويل من درجات حرارة في النظام المئوي إلى النظام الفهرنهيتي.

من الممكن استخدام الراسم xy أو الـ XY plotter لرسم العلاقة بين درجة الحرارة بالتدريج المئوي والتدريج الفهرنهيتي كما في شكل (٧,٢٧) عن طريق وضع مصدر دالة تصاعدية خطية Ramp function بدلا من المصدر الثابت Constant.

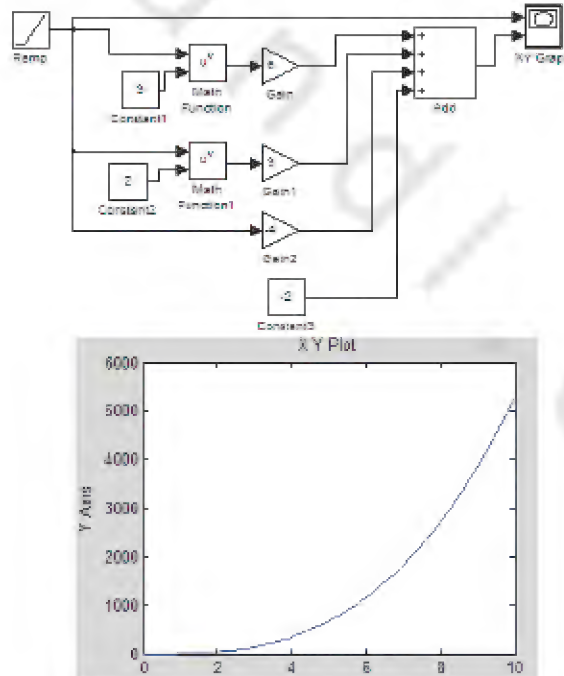


شكل (٧,٢٧). استخدام الـــ XY Plot لرسم العلاقة بين الدرجة المئوية والدرجة الفهرنهيتي.

يمكن أيضا محاكاة أي معادلة رياضية مثل المعادلة التالية :

$$f = 5x^3 + 3x^2 - 4x - 2$$

شكل (٧,٢٨) يبين محاكاة هذه المعادلة من خلال استخدام بلوك Math function أو الدالة الحسابية، ومن خلال خواصه نختار الدالة Pow حيث يقوم هذا البلوك بحساب الدخل الأول (الأعلى) مرفوعا إلى القوة الموجودة على الدخل الثاني (الأسفل). بلوك الدالة الحسابية الأول سيحسب الدخل x القادم من الدالة الخطية مرفوعا إلى القوة ٥. استخدمنا أيضا بلوك التجميع Add لجمع أكثر من دخل حيث بالنقر المزدوج على هذا البلوك يمكن زيادة عدد المداخل إلى العدد المطلوب عن طريق زيادة الإشارة (+) للدخل الذي سيدخل ليتم جمعه والإشارة (–) للدخل الذي سيتم طرحه. نتيجة محاكاة هذه المعادلة على الـ XY Plot موضحة في نفس الشكل (٧,٢٨).



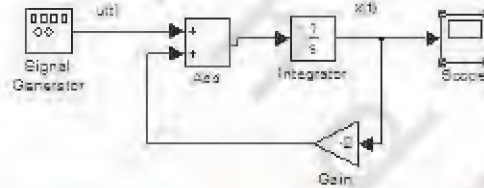
شكل (٧,٢٨). محاكاة معادلة حسابية من الدرجة الثالثة.

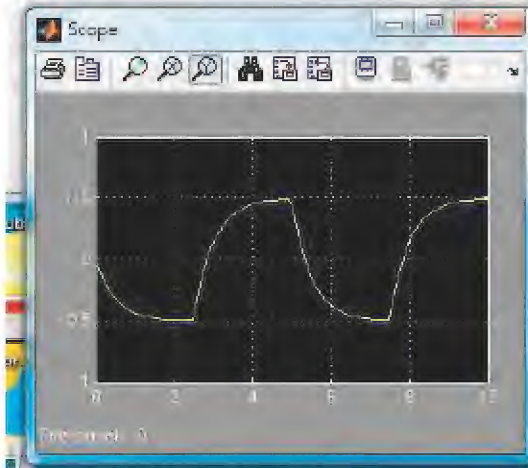
(٧,٦) محاكاة دوال العبور للأنظمة

دالة العبور لأي نظام من الممكن أن تكون من البساطة في صورة معادلة تفاضلية من الدرجة الأولى كالمعادلة التالية :

$$\frac{dx(t)}{dt} = -2x(t) + u(t)$$

هذه المعادلة تمثل نظام خرجه x(t) ودخله هو u(t)، حيث تعنى هذه المعادلة أن تفاضل الخرج زائد ضعف الخرج تساوي الدخل. أو بتكامل طرفي المعادلة فإن ذلك يعنى أن الخرج مضروبا في -٢ زائد الدخل u(t) ويتم تكاملهما معا سيعطي الخرج x(t). ولذلك يمكن محاكاة هذه الدالة كما في شكل (٧,٢٩). في هذا الشكل تم وضع الدخل في صورة موجة مربعة تحاكي دالة الخطوة حتى يكون خرج الأوسولوسكوب ممثلا لاستجابة الخطوة لهذا النظام.





شكل (٧,٢٩). محاكاة استجابة الخطوة لأحد الأنظمة البسيطة.

هناك في مكتبة الدوال الحسابية Math functions يوجد بلوك اسمه دالة العبور Transfer fcn حيث يمكن تمثيل أي دالة عبور (داخل هذا البلوك) في محول لابلاس Laplace transformation حتى يمكن دراسة أي استجابة لهذا النظام. لتمثيل أي معادلة تفاضلية في محول لابلاس نقوم بإجراء محول لابلاس على هذه المعادلة كما يلي:

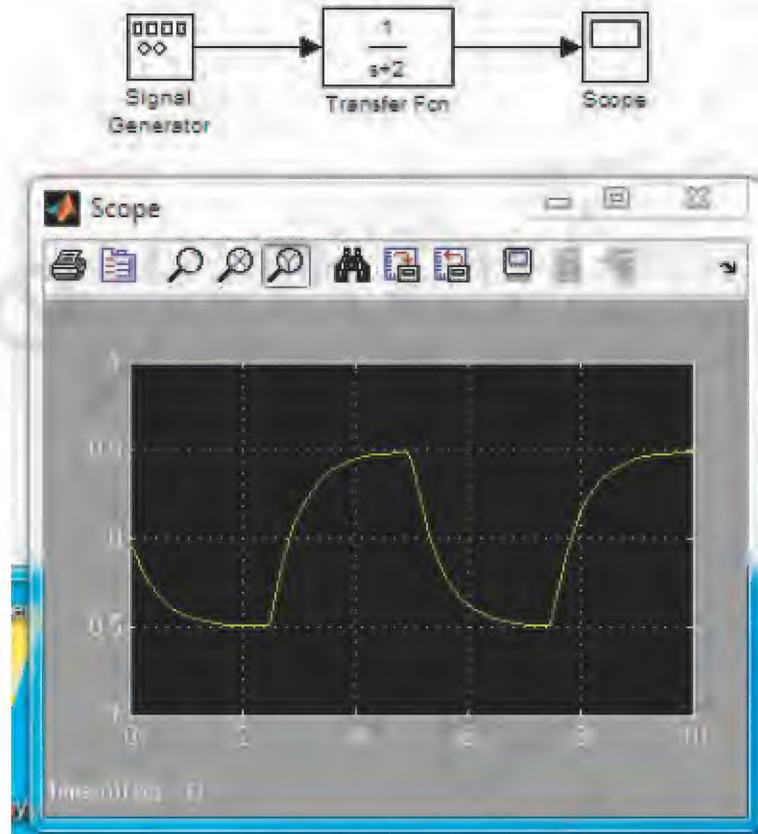
$$\frac{dx(t)}{dt} = -2x(t) + u(t)$$

بإجراء تحويل لابلاس لهذه المعادلة وإجراء بعض العمليات البسيطة نحصل على ما يلي:

$$Sx = -2x + u$$

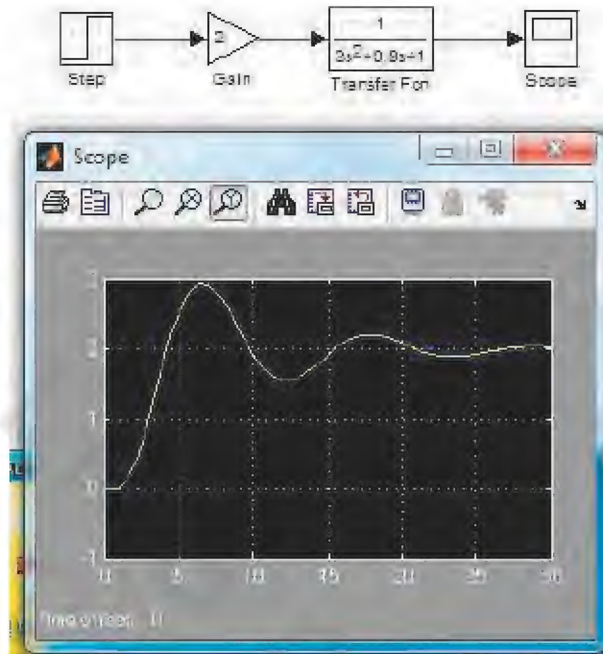
$$\frac{x}{u} = \frac{1}{S+2}$$

الآن يمكن تجميع البلوكات المبينة في شكل (٧،٣٠) والتي تحاكي هذا النظام. بلوك دالة العبور Transfer Fcn يمكن الحصول عليه من مكتبة سميولينك للدوال المستمرة Continuous. بالنقر مرتين على هذا البلوك وإدخال معاملات البسط وهي [١]، ومعاملات المقام وهي [١ ٢]، وضبط تردد إشارة الدخل والتنفيذ نحصل على الخرج على شاشة الأوسولوسكوب كما في شكل (٧،٣٠). لاحظ نفس الاستجابة كما في شكل (٧،٢٩). يجب أن نأخذ في الاعتبار أن الشرط الأولى Initial condition يساوي صفراً.

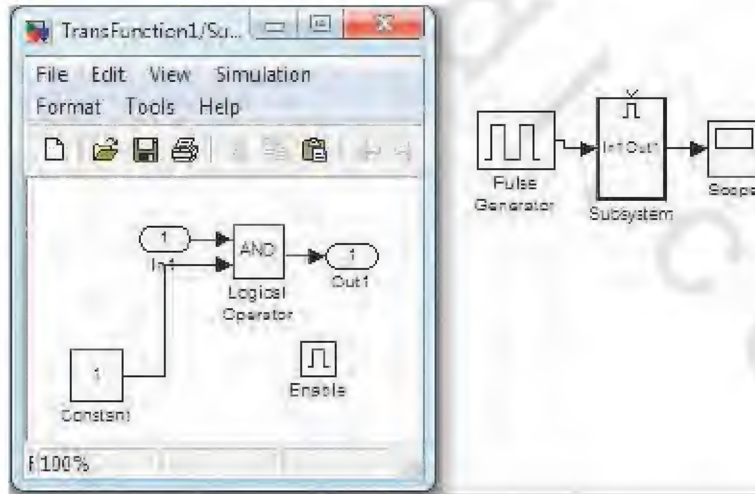


شكل (٧,٣٠). استجابة الخطوة باستخدام بلوك دالة العبور **Transfer Fcn** من مكتبة **Continuous**.

كمثال آخر سنقدم استجابة نظام تحكم من الدرجة الثانية كما في شكل (٧,٣١). حاول تغيير معاملات دالة العبور ومعامل التكبير لترى النتيجة على الأوسولوسكوب.



شكل (٧,٣١). استجابة الخطوة لنظام من الدرجة الثانية.



شكل (٧,٣٢). فتح نظام فرعي لإضافة طرف تنشيط له.

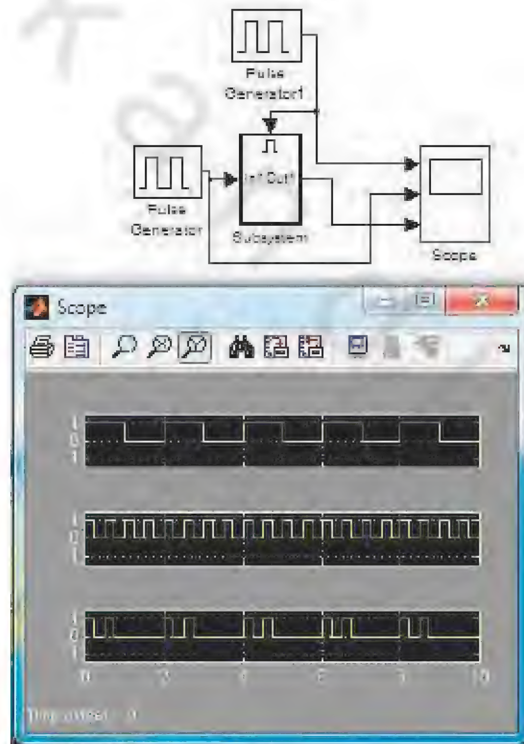
(٧,٧) تنشيط الأنظمة

في الكثير من التطبيقات نريد تشغيل النظام إذا كان أحد دخوله نشطا، ولا يعمل إذا كان هذا الدخل غير نشط. المقصود بكلمة نشط هنا أن هذا الدخل يكون موجباً أو أكبر من الصفر، وأما عدم النشاط فيعنى أن الدخل يكون سالباً أو صفراً. افترض أننا صممنا نظاماً فرعياً ونريد إضافة طرف تنشيط لهذا النظام الفرعي. العملية ببساطة هي أننا سنقوم بفتح هذا النظام الفرعي بالنقر عليه مرتين، ثم إضافة بلوك التنشيط Enable، ثم نغلق هذا النظام الفرعي مرة أخرى، حيث سنرى أنه تمت إضافة طرف تنشيط للنظام الفرعي.

في شكل (٧,٣٢) قمنا بتصميم نظام بسيط مكون من بوابة AND وجعلنا أحد هذه الأطراف يساوي واحداً من خلال البلوك Constant وأما الطرف الآخر فهو عبارة عن المدخل (١) وخرج البوابة على المخرج (١) كما في الشكل. بعد ذلك قمنا بوضع كل هذه المكونات في نظام فرعي كما سبق. بعد ذلك نقوم بفتح هذا النظام الفرعي بالنقر عليه مرتين، ثم نضيف لهذا النظام الفرعي البلوك enable، وهذا البلوك نحصل عليه من مكتبة سميولينك Ports&subsystems. بعد وضع هذا البلوك داخل النظام الفرعي نقوم بإغلاقه، حيث سنرى أنه تمت إضافة مدخل للنظام الفرعي يمكن تنشيطه من خلاله كما في الشكل (٧,٣٢).

لاحظ أن بلوك التنشيط enable لا يمكن إضافته إلا على نظام فرعي subsystem وهذا هو السبب في أننا في شكل (٧,٣٢) قمنا بتكوين نظام فرعي، ثم فتحنا هذا النظام ثم أضفنا البلوك enable ثم أغلقنا النظام الفرعي مرة أخرى. الآن بعد إضافة طرف التنشيط يمكن استخدامه حيث سنقوم في هذا المثال بتوصيل مصدر للنبضات على هذا الطرف. سنجعل عرض النبضة على هذا الطرف أكبر كثيرا من عرض النبضة

في مصدر النبضات الداخل على البوابة AND ثم ننفذ النظام، حيث سنرى أن النبضات الداخلة على البوابة AND لن تمر إلا في الفترات التي يكون فيها الطرف Enable يساوي واحداً كما في شكل (٧,٣٣). في شكل (٧,٣٣) الإشارة الأولى في خرج الأوسولوسكوب تمثل الإشارة الموجودة على الطرف enable والإشارة الثانية هي الإشارة الموصلة على دخل البوابة AND وأما الإشارة الثالثة فهي الإشارة الخارجة من البوابة AND حيث نرى تأثير طرف التنشيط enable.

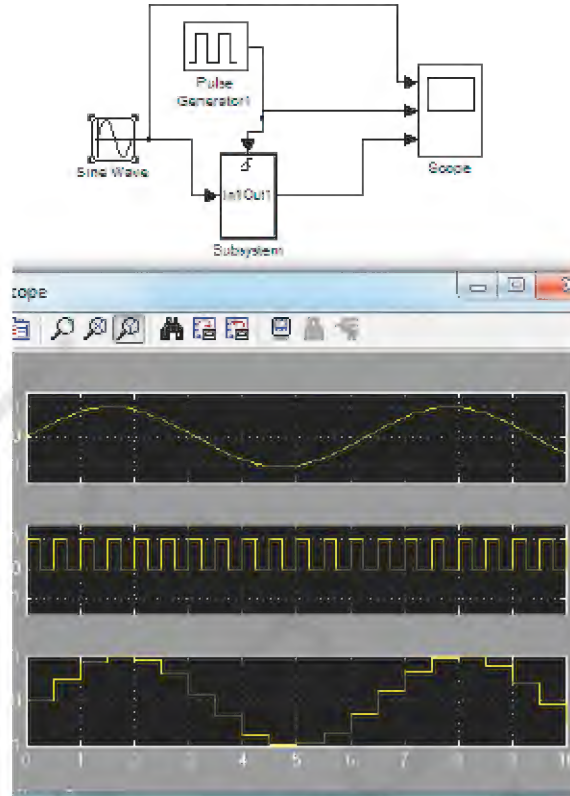


شكل (٧,٣٣). استخدام طرف التنشيط الذي تمت إضافته إلى النظام الفرعي.

(٧,٨) إضافة طرف قدح Trigger للأنظمة الفرعية

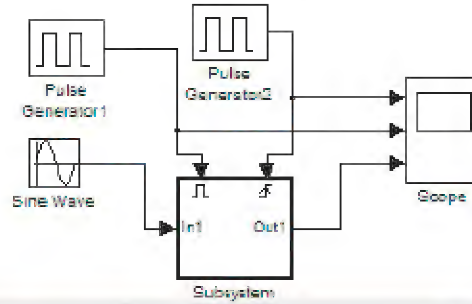
في الكثير من التطبيقات أيضا نحتاج لإضافة طرف قدح للنظام الفرعي. طرف القدح يختلف عن طرف التنشيط في أنه عند تنشيط طرف القدح، أي وضع إشارة عليه، فإنه يتم مسك خرج النظام عند هذه القيمة إلى أن تبدأ إشارة قدح أخرى، حيث تخرج قيمة الخرج الجديدة وهكذا. إنها بمثابة العينة والمسك Sample and hold المستخدمة في الكثير من الأنظمة. لإضافة طرف قدح لأي نظام نتبع نفس الخطوات التي تم شرحها تقريبا في إضافة طرف التنشيط، حيث يتم فتح النظام الفرعي بالنقر عليه مرتين، ثم نضيف بلوك القدح من مكتبة سميولينك Ports & Subsystems. ننقر مرتين على بلوك القدح لاختيار نمط القدح، حيث هناك ثلاثة أنماط للقدح أولها قدح مع الحافة الصاعدة للدخل الموجود على هذا الطرف، والنمط الثاني هو القدح مع الحافة النازلة، والنمط الثالث هو القدح إما مع الحافة الصاعدة وإما مع النازلة للإشارة الموجودة على طرف القدح. بعد ضبط نمط القدح نغلق النظام الفرعي ونبدأ في استخدامه كعنصر في النظام الكبير.

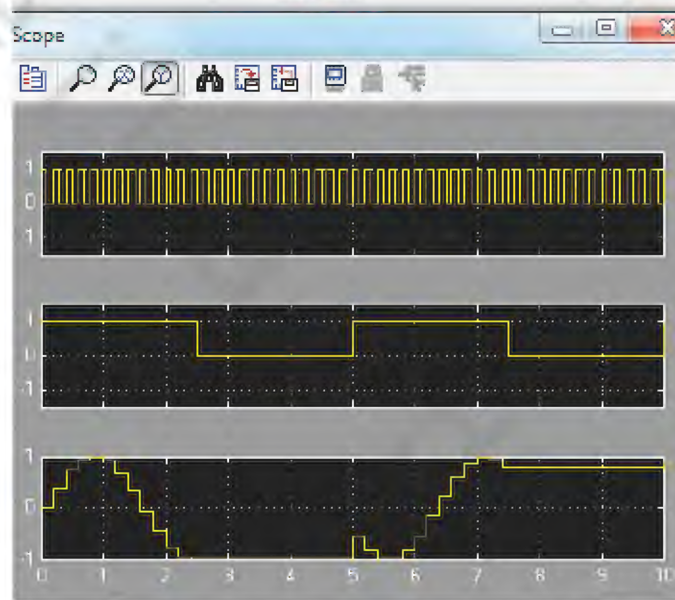
في شكل (٧,٣٤) تم عمل نظام فرعي مكون من بلوك ضرب Product من مكتبة سميولينك Commonly used blocks حيث تم وضع أحد أطرافه على بلوك المصدر Constant ليتم ضرب الطرف الآخر في واحد. ثم قمنا بإضافة بلوك القدح trigger إلى هذا النظام الفرعي بعد فتحه بالنقر عليه مرتين وضبط بلوك القدح ليتم القدح على الحافة الصاعدة. بعد ذلك قمنا بغلق النظام الفرعي. ثم استخدمنا هذا النظام الفرعي في بناء النظام الفرعي الموضح في شكل (٧,٣٤) حيث تم إدخال موجة جيبية على طرف الدخل للنظام الفرعي، واستخدام بلوك مولد نبضات على طرف القدح كما في الشكل. لاحظ من خرج الأوسولوسكوب كيف أنه مع الحافة الصاعدة في طرف القدح يتم مسك الخرج على قيمة الموجة الجيبية عند هذه اللحظة كما وضحنا سابقا.



شكل (٧،٣٤). إضافة طرف قدح trigger على النظام الفرعي.

لكي نرى الفرق بين تأثير طرفي القدح والتنشيط سنفتح النظام الفرعي السابق ونضيف له بلوك تنشيط وبلوك قدح (إذا حاولت إضافة بلوك تنشيط لنظام فرعي به بلوك قدح ربما لا يوافق النظام على ذلك، في هذه الحالة قم بإلغاء بلوك القدح ثم أضف بلوك التنشيط، ثم أضف بعده بلوك القدح مرة ثانية فسيتم قبوله). اضبط الترددات في المصادر المختلفة، حيث سنلاحظ أنه عندما يكون طرف التنشيط صفرا كما في خرج الأوسولوسكوب الثاني، فإن خرج النظام الفرعي يكون صفرا. وأما عندما يكون طرف القدح صفرا فإن خرج النظام الفرعي يكون هو قيمة الخرج عند لحظة القدح، وليس صفرا كما في حالة طرف التنشيط، وكما هو موضح في الشكل (٧،٣٥).

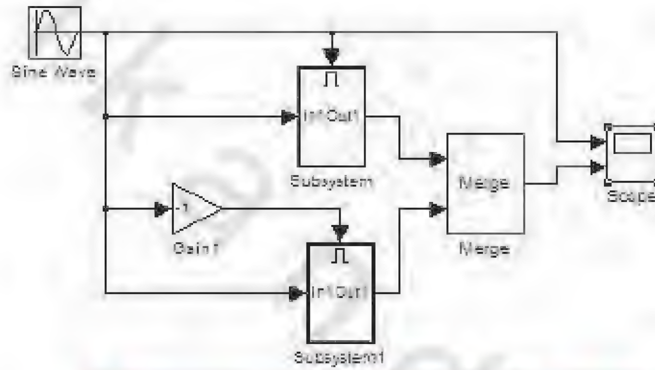


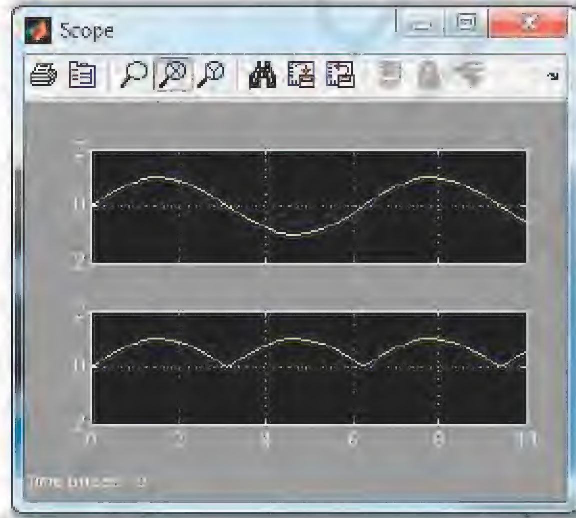


شكل (٧,٣٥). توضيح الفرق بين تأثير طرفي القدح والتنشيط.

شكل (٧,٣٦) يبن استخدام نظامين فرعيين مع طرف تنشيط لكل منهما للحصول على موجة جيبية كاملة التوحيد. النظام الفرعي العلوي يحتوي بلوك تكبير بمعامل تكبير يساوي واحداً وتم توصيل طرف تنشيطه بنفس الموجة الجيبية بحيث يتم تنشيط هذا النظام الفرعي في النصف الموجب للموجة فقط ؛ وبالتالي يتم تمرير هذا النصف الموجب للموجة إلى الأوسولوسكوب من خلال بلوك الدمج Merge. النظام

الفرعي الثاني (الأسفل) يحتوي بلوك تكبير Gain معامل تكبيره يساوي ١- ليعكس النصف السالب للموجة، وتم توصيل طرف التنشيط لهذا النظام الفرعي على بلوك الموجة الجيبية من خلال بلوك تكبير بمعامل تكبير-١ ليكون نشطا في النصف السالب من الموجة فقط. خرج النظام الفرعي تم تمريره إلى الأوسولوسكوب من خلال بلوك الدمج أيضا كما في شكل (٧,٣٦) الذي يبين خرج الأوسولوسكوب أيضا.





شكل (٧,٣٦). محاكاة موحد موجة كاملة.

بلوك الدمج Merge له عدة دخول وخرج واحد، ويقوم هذا البلوك بدمج كل هذه الدخول في خرج واحد، بحيث يكون الخرج مساويا للدخل القادم من البلوك الفعال أو الذي يعمل. هذا البلوك مأخوذ من مكتبة سميولينك Signal Routing.

سنكتفي بهذا القدر من الأمثلة والمواضيع الخاصة ببرنامج سميولينك؛ لأن السميولينك لكي يتم تغطيته بالكامل فإنه لن يكفيه هذا الكم في هذا الكتاب، ولكنه يحتاج لكتاب منفرد. ولكننا نعتقد أنه بهذا الكم من المعلومات عن السميولينك نكون قد كشفنا ظلمات هذا البرنامج، بحيث يستطيع القارئ بعد ذلك أن يعتمد على نفسه في تجربة باقي البلوكات المهمة في مجال تخصصه واستخدامها في هذه التطبيقات ونشير أيضا إلى المراجع [١٢ و ١٣].

الفصل الثامن

اكتساب البيانات

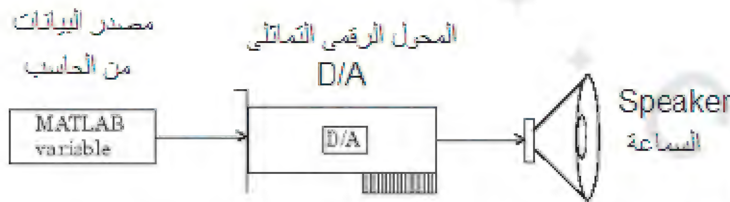
(٨,١) مقدمة

عملية إدخال وإخراج البيانات سواء التماثلية أو الرقمية من المواضيع ذات الأهمية للكثير من المهندسين في جميع التخصصات، حيث تكون هناك حاجة لإدخال هذه البيانات إلى الحاسب من خلال كارت اكتساب البيانات Data acquisition card الذي يتم تصنيعه عن طريق الكثير من الشركات المتخصصة في هذا المجال. أحد أمثلة هذه الكروت هو كارت الصوت المثبت داخل الحاسب، والذي سنستخدمه في الكثير من الأمثلة في هذا الفصل لإدخال وإخراج إشارة الصوت من وإلى الحاسب. نبدأ هذا الفصل بشرح للدوال المستخدمة في إخراج وإدخال البيانات التماثلية، ثم نتعرض لجهاز غاية في الأهمية و هو الأوسولوسكوب. بعد ذلك نعرض بلوكات السيميولينك التي تخص اكتساب البيانات مع شرح مفصل لاستخدام بلوك الإدخال. بعد إدخال الإشارة إلى الحاسب تكون هناك في العادة مرحلة معالجة لهذه البيانات التي تتم داخل الحاسب، وبالطبع فإن هذه المرحلة تعتبر خارج أهداف الكتاب. النصيحة هنا هي أنه على القارئ أن يرجع دائما إلى المساعدة help التي يوفرها ماتلاب، والتي تحتوي

الكثير من الدوال وخواص كل منها، والتي هي خارج نطاق الكتاب والتي لا يتسع المجال للشرح التفصيلي لها هنا، ولكننا اكتفينا هنا بتقديم القليل من الأفكار في المجال ووضع القارئ على الطريق ليستمر هو في الاستزادة إن أراد.

(٨,٢) إخراج البيانات التماثلية

سنشرح في هذا الجزء الدوال المستخدمة في إخراج بيانات تماثلية على أحد أجهزة إخراج البيانات التماثلية. أجهزة الإخراج هذه من الممكن أن تكون كروتاً يتم تثبيتها داخل علبة الحاسب أو أجهزة خارجية يتم توصيلها على أحد أطراف الحاسب، وهناك الكثير من الشركات التي تنتج مثل هذه الأجهزة وهذه الكروت. في هذا الفصل سنستخدم كارت الصوت المثبت داخل أي حاسب لتجربة إخراج البيانات عليه. سنعرض هنا الأوامر التي يمكن استخدامها لإخراج البيانات الرقمية التي يتم تحويلها إلى الصورة التماثلية عن طريق المحول الرقمي التماثلي D/A ومنه إلى السماعة مباشرة كما في شكل (٨,١) الذي يبين شكلا توضيحيا لهذه العملية.



شكل (٨,١). إخراج بيانات تماثلية من داخل الحاسب على كارت الصوت.

تقدم مكتبة اكتساب البيانات Data acquisition toolbox العديد من الأوامر التي تسهل عملية إخراج البيانات من داخل الحاسب إلى الخارج، وسنقدم في هذا الجزء شرحا لأهم هذه الأوامر.

لابد من إنشاء هدف يتم إخراج هذه البيانات عليه، وذلك من خلال الأمر analogoutput() الصورة العامة لهذا الأمر هي :

```
AO = analogoutput('adaptor')
AO = analogoutput('adaptor',ID)
```

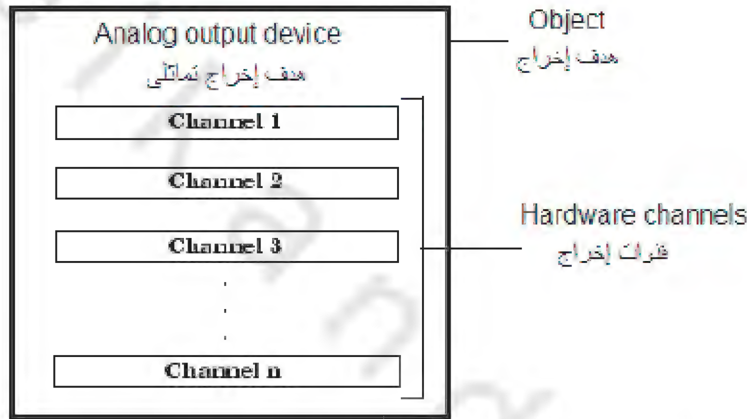
حيث AO هو اسم اختياري للهدف الذي سيتم إخراج البيانات عليه، و 'adaptor' هو اسم جهاز الإخراج (الكارت أو الـ hardware) الذي سيتم إخراج البيانات عليه. في حالة إخراج البيانات على كارت الصوت الملحق بالحاسب فإن الـ adaptor سيتم وضعه بالاسم 'winsound'. من الأجهزة أو الكروت التي يدعمها ماتلاب في هذا المجال الكروت hpe1432 و keithley و mcc و nidaq وذلك بجانب كارت الصوت الملحق بالحاسب. في الصورة الثانية من الأمر يتم وضع الرقم ID بجانب اسم الجهاز. هذا الرقم يكون غير ضروري في حالة التعامل مع كارت الصوت، أو يتم وضعه بصفر كما في البرنامج التالي. هذا الرقم يكون رقما صحيحا أو أحيانا يكون سلسلة أحرف. في حالة استخدام كارت شركة National Instruments يتم كتابة هذا الأمر كالتالي :

```
AO = analogoutput('nidaq','Dev1');
```

عند إنشاء هدف للإخراج مثل AO لا يتم فتح قناة لإخراج هذه البيانات من هدف الإخراج إلى جهاز الإخراج نفسه. لكي يتم ذلك نستخدم الأمر addchannel() يمكنك أن تتخيل الهدف الذي تقوم بإنشائه على أنه تانك أو محتوى تخرج منه هذه القنوات channels أو الخطوط، حيث القنوات تستخدم مع الإخراج التماثلي، والخطوط lines تستخدم مع الإخراج الرقمي. شكل (٨،٢) يبين رسما توضيحيا لمعنى

الهدف والقنوات الملحقة به. في حالة الإخراج الرقمي يتم استبدال القنوات بخطوط تمثل البيانات الرقمية. الأمران التاليان يبينان إنشاء هدف ao وفتح قناتين للإخراج التماثلي من خلال هذا الهدف.

```
ao = analoginput('winsound');
addchannel(ao,1:2)
```



شكل (٨,٢). رسم توضيحي لهدف الإخراج التماثلي والقنوات الملحقة به.

عند إضافة القنوات يجب أن تتذكر أن جميع القنوات يجب أن يكون لها نفس معدل أخذ العينات sampling rate ، ويجب ألا تضاف قنوات من أجهزة مختلفة في نفس الهدف.

بعد إنشاء الهدف وفتح القنوات فيها لابد من تحديد بعض الخواص لهذه القنوات. من هذه الخواص خاصية معدل أخذ العينات sampling rate ، والتي يتم تحديدها كما يلي :

```
ao = analogoutput('nidaq','Dev1');
addchannel(ao,0:1);
set(ao,'SampleRate',100000)
```

حيث تم ضبط هذا المعدل بالأمرset() والذي تم فيه تحديد الهدف وهو ao والخاصية 'SampleRate' التي تم تحديدها ب ١٠٠ كيلوهرتز. تذكر أنه لكل جهاز إخراج أو كارت من كروت إخراج البيانات التماثلية يكون هناك معدلات محددة لأخذ العينات مصاحبة لهذا الكارت يجب معرفتها مسبقا من كتالوج هذا الكارت. في أمر تحديد معدل العينات إذا كانت العينات المحددة في الأمر set() متوافقة مع أحد قيم العينات المحددة في كتالوج الكارت فلن تكون هناك مشكلة. إذا كان المعدل المحدد في الأمر set() لا يتوافق مع القيم المحددة في الكتالوج، ولكنه يقع في المدى المحدد لهذه المعدلات في الكتالوج، فإن ماتلاب سيقرب هذا المعدل لأقرب قيمة معدل محددة في كتالوج هذا الجهاز. أما إذا كانت القيمة المحددة بالأمر set() خارج المدى المحدد في كتالوج هذا الكارت، فإن ماتلاب سيعطي رسالة خطأ على ذلك.

بالنسبة لكروت الصوت الملحقة بالحاسب أقل معدل عينات هو ٨ كيلوهرتز، وأعلى معدل عينات هو ٤٤,١ كيلوهرتز، وقد تمتد حتى ٩٦ كيلوهرتز في بعض الكروت. القيم المحددة أو القياسية لمعدل العينات هي ٨ و ١١,٠٢٥ و ٢٢,٠٥ و ٤٤,١ كيلوهرتز.

الخاصية الثانية التي يجب تحديدها لكل هدف هي خاصية القدح trigger. هذه الخاصية تحدد توقيت إخراج البيانات على جهاز إخراج البيانات التماثلية الذي هو السماعة في حالة كارت الصوت. هناك نوعان شائعان من القدح يتم اختيار أحدهما بالأمر TriggerType. النوع الأول من القدح هو Immediate والذي يبدأ إخراج البيانات بمجرد تنفيذ الأمر start. النوع الثاني من القدح هو اليدوي Manual والذي يبدأ إخراج البيانات بعد تنفيذ الدالة trigger من مساحة العمل workspace. البرنامج التالي يبين كل هذه العمليات، بمجرد تنفيذ هذه الصورة من البرنامج ستسمع صفارة سميكة من سماعة الحاسب مباشرة.

```
%Output analog data to the sound card
ao = analogoutput('winsound');
addchannel(ao,1:2);
set(ao,'SampleRate',44100)
data = sin(linspace(0,2*pi*500,44100)');
putdata(ao,[data data])
start(ao)
```

في الصورة التالية من البرنامج تم إضافة أمر لتحديد طريقة القدح بأن تكون يدوية ، ولذلك فإنه بتنفيذ البرنامج فلن تسمع الصوت إلى أن تكتب الدالة trigger(ao) في مساحة العمل ، حيث بعدها ستسمع الصوت.

```
%Reading analog data from the sound card
ao = analogoutput('winsound');
addchannel(ao,1:2);
set(ao,'SampleRate',44100)
set(ao,'TriggerType','Manual');
data = sin(linspace(0,2*pi*500,44100)');
putdata(ao,[data data])
start(ao)
```

في البرنامج السابق البيانات data التي سيتم إخراجها عبارة عن الدالة الجيبية sin() لمجموعة من نقط البيانات المولدة بالأمر linspace() هذا الأمر يولد متجه من البيانات عددها ٤٤١٠٠ بين القيمة صفر والقيمة 2*pi*500 بحيث تكون المسافة أو الزمن بين هذه النقاط خطية. لاحظ أن مدة هذا الصوت ستكون ثانية واحدة ، حيث إن عدد النقاط يساوي معدل العينات في الثانية المحدد بأمر تحديد معدل العينات. يمكن التحكم في تردد الصوت الناتج عن طريق تغيير الرقم ٥٠٠ حيث بزيادة هذا الرقم سيزيد التردد الناتج.

بعد تجهيز البيانات بالأمر data يجب وضعها في طابور على جهاز الإخراج عن طريق الأمر putdata() كما في البرنامج السابق. لاحظ أن هدف الإخراج ao يحتوي

قناتين ٢ : ١ كما في أمر إضافة القنوات addchannel() في البرنامج السابق. لذلك عند وضع البيانات على جهاز الإخراج لابد من وضعها على كل قناة تم فتحها في هذا الهدف، لذلك تم وضع البيانات في صورة عمودين [data data] كما في البرنامج. يمكنك التجربة مع قناة واحدة أو أكثر من قناة.

بعد أن تم وضع البيانات في صورة طابور أمام كل قناة يمكنك أن تنفذ البرنامج بالأمر start() كما في البرنامج، حيث بمجرد تنفيذ هذا الأمر ستسمع الصوت مباشرة إذا لم يكن القدح يدويا. تذكر أنه لابد من تنفيذ هذا الأمر لكي تسمع الصوت.

إذا كان زمن إخراج الإشارة على جهاز الإخراج طويلا يمكنك إيقافه في أي وقت عن طريق الأمر stop().

(٨,٣) إدخال البيانات التماثلية

تقدم مكتبة اكتساب البيانات Data acquisition toolbox العديد من الأوامر التي تسهل عملية إدخال البيانات من خارج الحاسب إلى داخله. مثلا يمكن إدخال إشارة قادمة من أي حساس، وليكن حساس الحرارة إلى الحاسب وتخزينها في الذاكرة أو إجراء بعض المعالجات عليها. سنقدم في هذا الجزء شرحا لأهم هذه الأوامر. إن أوامر إدخال البيانات للحاسب لن تختلف كثيرا عن أوامر إخراج البيانات من الحاسب التي سبق شرحها في الجزء السابق، ولذلك نعتقد أن العملية هنا ستكون أسهل.

أول خطوات إدخال البيانات تكون عن طريق إنشاء هدف إدخال كما فعلنا في حالة إخراج البيانات بالأمر التالي الذي سنستخدم معه كارت الصوت الملحق بالحاسب:

```
ai = analoginput('winsound');
```

في حالة استخدام الكارت المصنع من شركة National Instrument مثلا يمكن استخدام الأمر التالي :

```
ai = analoginput('nidaq','Dev1');
```

بمجرد فتح هدف الإدخال يقوم ماتلاب بتحديد اسمه ونوعه بحيث يمكنك السؤال عنهما كما يلي :

```
>> ai = analoginput('winsound');
>> get(ai,{'Name','Type'})
ans =
    'winsound0-AI'    'Analog Input'
```

في الأوامر السابقة تم فتح الهدف بالأمر analoginput() ، وبعد ذلك تم السؤال عن هذا الهدف بالأمر get() للسؤال عن اسم هذا الهدف ونوعه فكانت الإجابة أن اسم هذا الهدف هو winsound0-AI ونوعه أنه هدف إدخال بيانات Analog Input.

كما في حالة إخراج البيانات فإنه في حالة إدخال البيانات لابد من فتح قنوات بعد إنشاء الهدف. في حالة كارت الصوت الملحق بالحاسب يمكن فتح قناة واحدة أو قناتين. في حالة فتح قناة واحدة فإن نمط الصوت في هذه الحالة سيكون أحاديًّا mono وفي حالة فتح قناتين سيكون نمط الصوت ثنائياً أو مجسماً stereo. الأوامر التالية تستخدم في فتح هذه القنوات :

```
addchannel(ai,1);
```

وذلك لفتح قناة واحدة في الهدف ai في كارت الصوت. الأمر التالي سيفتح القناة الثانية في نفس الهدف السابق :

```
addchannel(ai,2);
```

ويمكن فتح القناتين بنفس الأمر كما يلي :

```
addchannel(ai,1:2);
```

يمكنك حذف أي واحدة من القناتين بالأمر التالي :

```
delete(ai.Channel(2))
```

عندما تكون في النمط الثنائي وتريد حذف قناة ، فلابد أن تكون هذه القناة هي القناة الثانية ، إذا حذفت القناة الأولى وتركت الثانية فإن ماتلاب سيعطي رسالة خطأ على ذلك.

هناك خواص لجهاز (كارت) الإدخال لابد من ضبطها، تماما كما فعلنا مع أجهزة الإخراج. من هذه الخواص معدل أخذ العينات sampling rate والذي يمكن ضبطه كما في الأوامر التالية :

```
ai = analoginput('nidaq','Dev1');
addchannel(ai,0:1);
set(ai,'SampleRate',100000)
```

كما ذكرنا مسبقا أن معدل العينات في الأوامر السابقة يجب أن يكون واقعا ضمن المدى المحدد عن طريق الشركة المنتجة لهذا الجهاز. إذا لم تكن القيمة المحددة في الأمر متوافقة مع القيم المحددة في كتالوج الكارت، فإن ماتلاب يساويها بأقرب قيمة في القيم المحددة في كتالوج الكارت، ويمكن الاستفسار عن القيمة الحقيقية التي تم اعتبارها بالمتلاب كما في الأمر التالي :

```
ActualRate = get(ai,'SampleRate');
```

الخاصية الثانية هي خاصية القدح. وهي كما شرحنا مسبقا إما أن يكون القدح فوريا immediate حيث تحدث عملية إدخال البيانات بعد تنفيذ الأمر start فورا، وإما يدويا حيث تحدث عملية القدح بعد تنفيذ أمر القدح trigger يدويا. أيضا يمكن أن تحدث عملية القدح برمجيا بمجرد أن تتحقق شروط عملية القدح في البرنامج. الأمر التالي يوضح مثالا على طريقة القدح :

```
set(ai,'TriggerType')
```

هناك أيضا الخاصية SamplesPerTrigger التي تحدد عدد العينات التي يتم اكتسابها مع كل عملية قدح كما في المثال التالي :

```
set(ai,'SamplesPerTrigger',500000)
```

حيث سيتم أخذ ٥٠٠٠٠٠ عينة أي ما يعادل زمن مقداره خمس ثوانٍ بفرض أن معدل أخذ العينات هو ١٠٠٠٠٠ عينة في الثانية. في بعض التطبيقات نحتاج أن يكون عدد العينات التي يتم قراءتها لانهائيا، أي أننا نريد أن تكون عملية اكتساب البيانات مستمرة ولا تتوقف، ويمكن عمل ذلك كما في المثال التالي :

```
set(ai,'SamplesPerTrigger',inf)
```

في هذه الحالة ستستمر عملية اكتساب البيانات إلى يتم تنفيذ الأمر و الدالة stop.

المثال التالي سيقرأ بيانات من خلال الميكروفون الملحق بكارت الصوت، ثم نقوم بإخراج هذه البيانات على السماعة لنسمعها.حيث تم ضبط معدل العينات على ٨كيلوبايت، وتم تحديد عدد العينات المطلوب قراءتها بواحد ثانية في معدل العينات، وتم ضبط القدح ليكون يدويا بحيث تبدأ عملية القراءة فور تنفيذ الأمر trigger(AI).أمر الانتظار wait تم تنفيذه حتى يتم الانتظار لحين الانتهاء من عملية القراءة للبيانات، وتم زيادة هذا الزمن بمقدار واحد ثانية إضافية حتى يتم التأكد من الانتهاء من عملية القراءة. في آخر عملية القراءة تم استخدام الأمر getdata(AI) لوضع البيانات المقروءة في المصفوفة data تمهيدا لإخراجها على السماعة في الجزء الثاني من البرنامج. تذكر أن يكون معدل العينات في حالة القراءة مساويا لمعدل العينات في إخراج هذه البيانات.

```
%Reading analog data from a mic and output to
loudspeaker on the soundcard
AI = analoginput('winsound');
chan = addchannel(AI,1);
duration = 1; %1 second acquisition
set(AI,'SampleRate',8000);
ActualRate = get(AI,'SampleRate');
set(AI,'SamplesPerTrigger',duration*ActualRate)
set(AI,'TriggerType','Manual')
start(AI);
trigger(AI);
wait(AI,duration + 1);
data = getdata(AI);

ao = analogoutput('winsound');
addchannel(ao,1:2);
set(ao,'SampleRate',8000)
putdata(ao,[data data])
start(ao)
```

بعد الانتهاء من عملية القراءة يمكنك التخلص من هذه البيانات التي تم قراءتها وإغلاق هدف القراءة المفتوح باستخدام الأمرين التاليين:

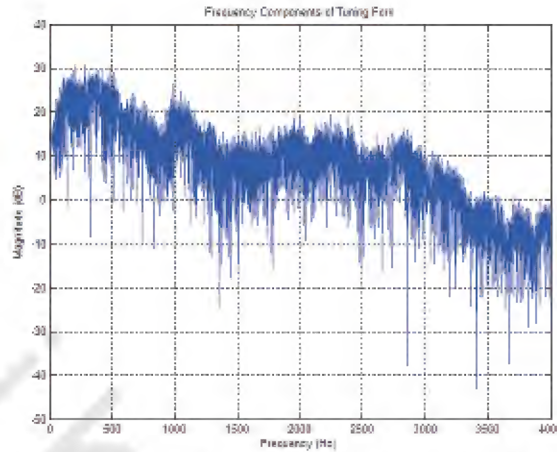
```
delete(AI)
clear AI
```

إن البيانات التي تم تسجيلها في المصفوفة data في البرنامج السابق يمكن الاستفادة منها في الكثير من التطبيقات وليس بغرض سماعها فقط. فمثلا يمكن إجراء محول فورير على هذه البيانات بحيث نتمكن من رؤية المحتويات الترددية فيها. لقد وفر ماتلاب الدالة التالية والمعدة خصيصا لتعطينا محول فورير للبيانات التي تم اكتسابها:

[f,mag] = daqdocfft(data,Fs,blocksize);

معاملات هذه الدالة هي المصفوفة data التي تحتوي البيانات التي تم اكتسابها، ومعدل العينات Fs أو تردد العينات، وعدد البيانات المكتسبة المراد حساب محول فورير لها. هذه الدالة daqdocfft تعطي العلاقة بين المقدار mag والتردد f اللذين يمكن رسمهما كما في البرنامج التالي وكما في شكل (٨,٣) الذي يوضح نتيجة محول فورير للصوت المدخل.

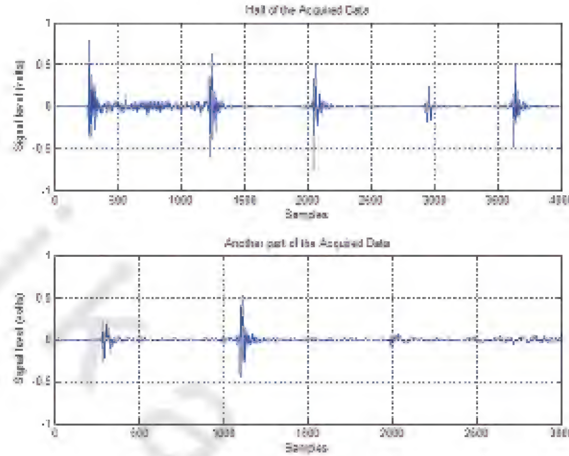
```
%Reading analog data and drawing its fourier transform
using the soundcard
AI = analoginput('winsound');
chan = addchannel(AI,1);
duration = 1; %1 second acquisition
set(AI,'SampleRate',8000);
ActualRate = get(AI,'SampleRate');
set(AI,'SamplesPerTrigger',duration*ActualRate)
set(AI,'TriggerType','Manual')
start(AI);
trigger(AI);
wait(AI,duration + 1);
data = getdata(AI);
[f,mag] = daqdocfft(data,Fs,blocksize);
xfft = abs(fft(data));%because fft is a complex quantity
% Avoid taking the log of 0.
index = find(xfft == 0);
xfft(index) = 1e-17;
mag = 20*log10(xfft);
mag = mag(1:floor(blocksize/2));
f = (0:length(mag)-1)*Fs/blocksize;
f = f(:);
plot(f,mag)
grid on
ylabel('Magnitude (dB)')
xlabel('Frequency (Hz)')
title('Frequency Components of Tuning Fork')
```



شكل (٨.٣). نتيجة محول فورير للبيانات المكتسبة من كارت الصوت.

الدالة data=getdata(AI) في المثال السابق تقرأ البيانات المكتسبة من خلال هدف الإدخال AI ويضعها في المصفوفة data. بمجرد الحصول على هذه البيانات في المصفوفة يمكن توظيفها في أي غرض، ومن ذلك حساب المحتويات الترددية كما رأينا في المثال السابق، أو يمكن رسمها مع الزمن كما سنرى في البرنامج التالي. لاحظ أنك يمكنك قراءة أي عدد من البيانات الموجودة في هدف الإدخال عن طريق تحديد عدد العينات التي تريدها في الدالة السابقة كما في الدالة data=getdata(AI,4000) حيث هذه الدالة ستقرأ ٤٠٠٠ عينة فقط من هدف الإدخال وتضعها في المصفوفة data. أي أنه إذا لم ينص على عدد العينات في الدالة getdata فإن عددها يكون كل العينات الموجودة في الهدف والمحددة بالخاصية SamplesPerTrigger والتي تحدد عدد العينات المكتسبة بعد كل عملية قدح. لاحظ أيضا أنه بعد تنفيذ الدالة data=getdata(AI,samples) فإن خاصية العينات المتاحة SamplesAvailable يتم إنقاصها بمقدار البيانات التي تم اكتسابها بالدالة getdata. إذا زاد عدد العينات في الدالة getdata عن العينات المتاحة في هدف الإدخال، فإن ماتلاب سيعطي إشارة خطأ. انظر البرنامج التالي الذي يرسم من الـ ٨٠٠٠ عينة

المتاحة أول ٤٠٠٠ عينة ثم الـ ٣٠٠٠ عينة التالية. حاول تنفيذ البرنامج وجرب أكثر من قيمة للعينات. شكل (٨,٤) يبين رسم هذه العينات مع الزمن.



شكل (٨,٤). رسم العينات المكتسبة مع الزمن.

```
%Reading analog data and drawing it using the soundcard
AI = analoginput('winsound');
chan = addchannel(AI,1);
duration = 1; %1 second acquisition
set(AI,'SampleRate',8000);
ActualRate = get(AI,'SampleRate');
set(AI,'SamplesPerTrigger',duration*ActualRate)
set(AI,'TriggerType','Manual')
%blocksize = get(AI,'SamplesPerTrigger');
%Fs = ActualRate;
start(AI);
trigger(AI);
wait(AI,duration + 1);
data = getdata(AI,4000);
subplot(211), plot(data), grid on
title('Half of the Acquired Data')
xlabel('Samples')
ylabel('Signal level (volts)')
data = getdata(AI,3000);
subplot(212), plot(data), grid on
title('Another part of the Acquired Data')
xlabel('Samples')
ylabel('Signal level (volts)')
```

الأمران AI.SamplesAvailable تم وضعهما قبل كل أمر getdata كما في البرنامج السابق وكانت نتيجتهما في شاشة نافذة الأوامر command window كما يلي ، مما يؤكد أنه بعد كل عملية اكتساب للبيانات يتم خصم هذه البيانات من الخاصية SamplesAvailable كما ذكرنا.

```
ans =
     8000
ans =
     4000
```

(٨,٤) الأوسولوسكوب

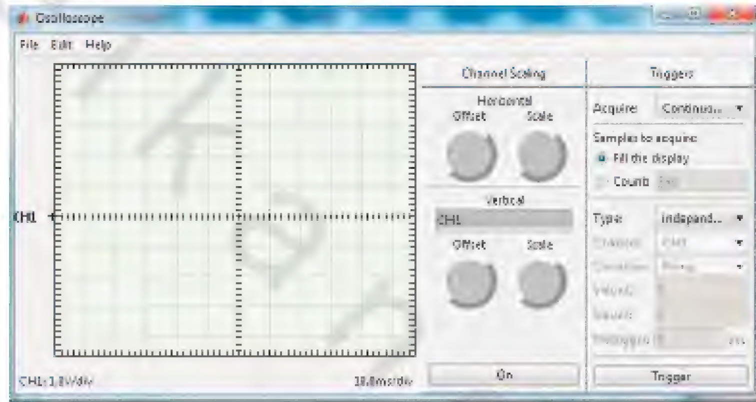
يوفر ماتلاب أوسولوسكوب يمكنك من خلاله عرض الإشارات التي تتعامل معها سواء المخرجة أو المدخلة. الأوامر التالية تفتح هدف إدخال، وتفتح فيه قناة، ثم بالأمر softscope يتم فتح الأوسولوسكوب كما في الشكل (٨,٥) :

```
>> ai=analoginput('winsound');
>> addchannel(ai,1);
>> softscope(ai)
```

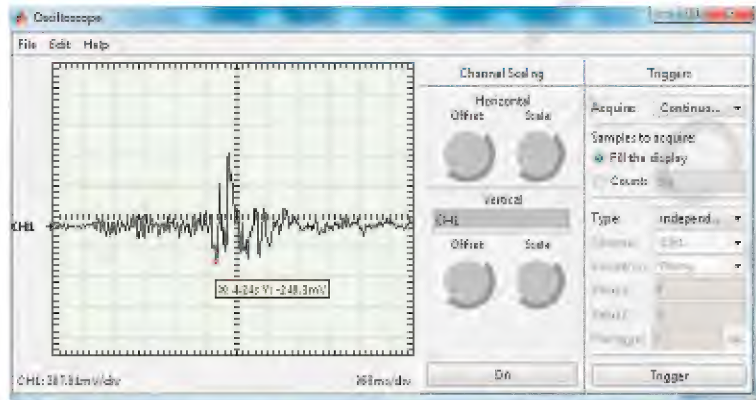
من شاشة الأوسولوسكوب وباختيار قائمة التحرير Edit ، ثم من قائمة التحرير يمكنك اختيار Hardware حيث ستفتح شاشة يمكن من خلالها ضبط الكثير من المعاملات ومنها الهدف أو الجهاز الذي سيتعامل مع الأوسولوسكوب والقناة التي سيتم عرضها وكذلك معدل العينة وغير ذلك من المعاملات. يمكن أيضا تغيير معدل العينة من خلال قائمة التحرير Edit مباشرة.

بالضغط على زر القدح Trigger يبدأ عرض الإشارة على الأوسولوسكوب ، ويبدأ تنشيط الأوسولوسكوب كما في شكل (٨,٦). يمكنك ملاحظة إشارة الصوت حيث بمجرد الضغط على زر القدح Trigger سترى إشارة الصوت التي يلتقطها الميكروفون ، وسترى تأثير صوتك عند الكلام أمام الميكروفون. يمكنك الآن استكشاف

أزرار الإزاحة Offset سواء الأفقية أو الرأسية عن طريق الضغط على الزر الأيسر للفأرة مع الحركة حيث سترى إزاحة خط الإشارة معك. يمكنك أيضا تغيير مسطرة الجهد Scale التي تحدد كم فولت لكل بعد رأسي على الشاشة فحاول تجربة ذلك. أيضا بالوقوف بالفأرة على أي نقطة على الإشارة سيعرض لك ماتلاب إحداثيات هذه النقطة من حيث الزمن عند هذه النقطة ومقدار الإشارة بالفولت كما في شكل (٨،٦) .



شكل (٨،٥). إظهار الأوسولوسكوب.



شكل (٨،٦). الأوسولوسكوب أثناء عرض إشارة الدخل وبيان إحداثيات إحدى النقط.

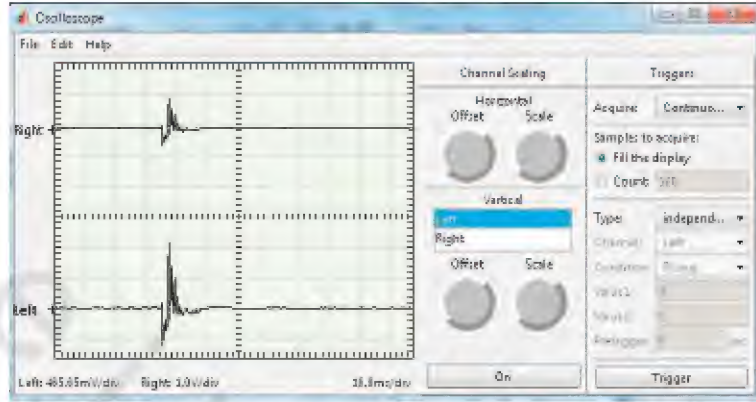
كما ذكرنا من قبل، فإن كارت الصوت يمكنه أن يحتوي على قناة أو اثنتين فقط، وفي حالة فتح القناتين معا بأمر addchannel فإن الأوسولوسكوب سيعرض القناتين كما في البرنامج التالي وكما هو مبين في شكل (٨،٧) :

```
>> ai=analoginput('winsound');
>> addchannel(ai,1:2);
>> softscope(ai)
```

في شكل (٨،٧) بمجرد الضغط على زر القدح سيبدأ عرض إشارة القناتين، حيث ستعرض القناة الأولى الإشارة اليمنى right وستعرض القناة الثانية الإشارة اليسرى left لأن كارت الصوت في هذه الحالة يعمل في النمط الثنائي أو المجسم أو الاستيريو stereo، مع العلم أن هذا التخصيص يكون تلقائيا.

في شكل (٨،٧) لاحظ وجود زراري التحكم left و right، على يمين شاشة العرض، في حالة النقر على الزر left فإن أزرار التحكم الخاصة بالإزاحة ومقدار الجهد ستعمل مع القناة الأولى، وفي حالة النقر على الزر right فإن أزرار التحكم ومقدار الجهد ستعمل مع القناة الثانية.

في حالة استخدام أجهزة الإدخال الخارجية من شركات أخرى، والتي تحتوي على العديد من القنوات التي يمكن فتحها من خلال تلك الأهداف، فإنه في هذه الحالة يمكن عرض كل قناة على قناة خاصة من قنوات الأوسولوسكوب.



شكل (٨.٧). عرض قناتي كارت الصوت.

استخدام بلوكات سميولينك الخاصة باكتساب البيانات

تحتوي مكتبة بلوكات سميولينك الخاصة باكتساب البيانات على البلوكات التالية :

• بلوك الدخل التماثلي Analog Input : الذي يكتسب بيانات قناة أو أكثر من قنوات جهاز الدخل التماثلي.

• بلوك الدخل التماثلي (عينة واحدة)Analog Input (Single Sample) : الذي يكتسب عينة واحدة من قناة أو أكثر من قنوات الدخل التماثلي.

• بلوك الإخراج التماثلي Analog Output : الذي يخرج بيانات تماثلية على قناة أو أكثر من قنوات جهاز لإخراج البيانات التماثلية.

• بلوك الإخراج التماثلي (عينة واحدة) Analog Output (Single Sample) : الذي يخرج عينة واحدة على قناة أو أكثر من قنوات جهاز لإخراج البيانات التماثلية.

• بلوك الإدخال الرقمي Digital Input : يدخل آخر مجموعة من القيم الموجودة على مجموعة من الخطوط الرقمية لجهاز من أجهزة الدخل الرقمي.

• بلوك الإخراج الرقمي Digital Output : الذي يخرج بيانات على عدد من خطوط البيانات الرقمية في جهاز لإخراج البيانات الرقمية.

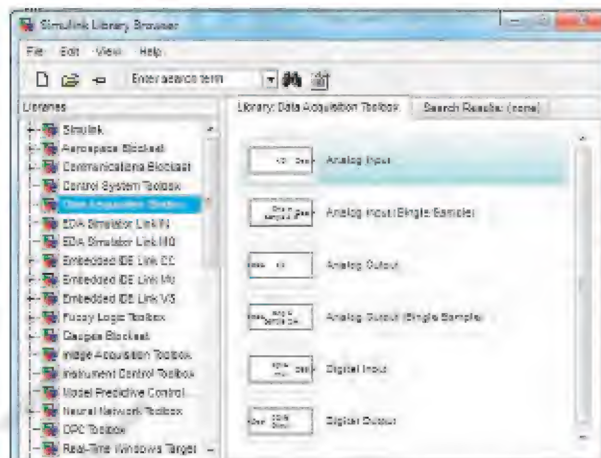
وسنتحدث هنا في هذا الجزء عن أحد هذه البلوكات، وهو بلوك الإدخال التماثلي ونستخدمه في مثال توضيحي لعرض إشارة صوت.

يمكن الدخول على مكتبة بلوكات اكتساب البيانات بطريقتين :

الأولى : من نافذة الأوامر لماتلاب اكتب الأمر daqlib حيث سينقلك ماتلاب إلى السميولينك ومنه إلى مكتبة اكتساب البيانات.

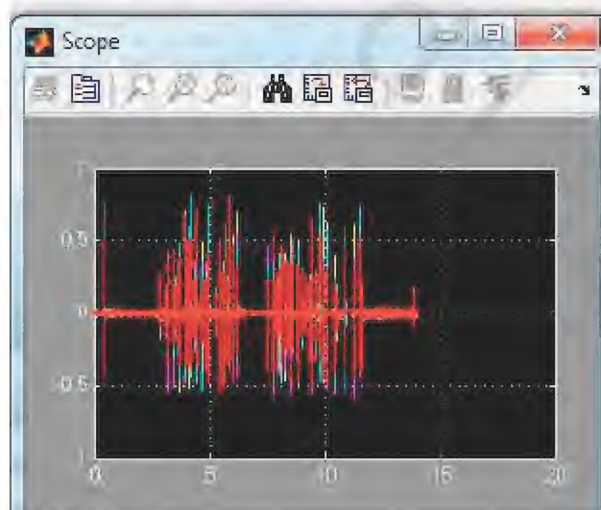
الثانية : بالنقر على أيقونة السميولينك والدخول فيه انقر على مجموعة بلوكات اكتساب البيانات Data acquisition toolbox حيث ستظهر مجموعة البلوكات التي ذكرناها سابقا والمبينة في شكل (٨,٨).

بعد الدخول على مكتبة بلوكات اكتساب البيانات يمكنك الآن فتح موديل جديد وإضافة بلوك إدخال تماثلي، ثم إضافة بلوك أوسولوسكوب وتوصيله على خرج بلوك الإدخال التماثلي لعرض إشارة الصوت التي سيقرأها البلوك من كارت الصوت من خلال الميكروفون كما في شكل (٨,٩). في شكل (٨,٩) انقر مرتين على بلوك الإدخال، حيث ستظهر أمامك شاشة تبين العديد من خواص هذا البلوك، أترك كل شيء كما هو فيما عدا الخاصية block size غيرها من ١ إلى ٥ وهي تعنى عدد العينات التي يتم أخذها في كل مرة قراءة من الكارت. أيضا غير زمن العرض إلى ٢٠ ثانية بدلا من ١٠ التي تمثل القيمة التلقائية، ثم ابدأ البرنامج وتكلم أمام الميكروفون حيث سترى صوتك على شاشة الأوسولوسكوب بعد النقر عليه مرتين ليفتح لك هذه الشاشة.



شكل (٨,٨). فتح مكتبة بلوكات اكتساب البيانات.





شكل (٨,٩). عرض إشارة الخرج المقروءة من الميكروفون الملحق بكارت الصوت.

سنكتفي بهذا القدر، حيث التعامل مع باقي البلوكات مشابه تماما لما قدمناه في بلوك الإدخال في الجزء السابق، فقط عليك النقر على كل بلوك ستستخدمه لتظهر لك خواص هذا البلوك فحاول أن تعدل فيها ما شئت لتتناسب مع التطبيق الذي تتعامل معه، وفي حالة أي صعوبة عليك اللجوء إلى المساعدة help الموجودة في ماتلاب.

الفصل التاسع

الحسابات الرمزية في برنامج ماتلاب

(٩,١) مقدمة

لقد تعودنا على التعامل الرقمي مع برنامج ماتلاب فمثلا يمكننا استخدام الماتلاب ، في حساب قيمة دالة مثل الدالة cos(x) أو الدالة sin(x) حيث المتغير x يؤول إلى قيمة معينة بالدرجات. الجديد هنا أننا يمكننا أن نطلب من ماتلاب حساب تفاضل أو تكامل أي واحدة من هذه الدوال فيعطينا الدالة cos(x) كتفاضل للدالة sin(x). كل هذا يتم من خلال صندوق أدوات Tool box خاص بالحسابات الرمزية ويسمى Symbolic Math Tool Box الذي سنقدم شرحا له في هذا الفصل. حيث يستخدم الماتلاب لإجراء بعض العمليات غير الرقمية على الدوال الرمزية مثل عمليات التفاضل والتكامل والنهايات وجمع المتواليات. سنرى كذلك كيفية حل المعادلات في عدد من المجاهيل بسهولة ويسر بدلا من الطرق التي استخدمناها سابقا كالبرمجة أو استخدام المصفوفات. إن التعامل مع المتغيرات الرمزية سيفتح مجالا واسعا وجديدا من مجالات استخدام الماتلاب ، وهذا التطبيق مفيد جدا ليس فقط للمهندسين ، ولكن لكل من يتعامل مع الرياضيات.

إن صندوق أدوات الحساب الرمزي يستخدم نوعا من المتغيرات الخاصة تسمى المتغيرات الرمزية symbolic variables التي ترمز للمتغير بسلسلة أحرف يتم حفظها في هذا المتغير. المثال التالي يوضح الفرق بين متغيرات ماتلاب الحسابية العادية مثل المتغيرات متضاعفة الدقة double والمتغيرات الرمزية. الأوامر التالية تبين نسخة من نافذة الأوامر command window في ماتلاب، حيث طلبنا الجذر التربيعي للرقم الحسابي ٢ بالأمر sqrt(2) فكانت الإجابة هي ١,٤١٤٢ ، أما عندما تم تعريف الرقم ٢ على أنه ثابت رمزي symbolic باستخدام الأمر sym(2) فكانت الإجابة هي ٢^(١/٢) حيث العلامة ^ تعنى الأس، مما يعنى أن المتغير $a = \sqrt{2}$. وعلى ذلك أصبح الرمز $\sqrt{2}$ محفوظا في المتغير a كرمز بدون حساب قيمته.

```
>> sqrt(2)
ans =
   1.4142
>> a=sqrt(sym(2)
a =
2^(1/2)
```

يمكن استدعاء قيمة a الحسابية مرة ثانية عن طريق تحويل المتغير a من صورته الرمزية إلى صورته المتضاعفة، وذلك باستخدام الأمر double(a) كما يلي :

```
>> double(a)
ans =
1.4142
```

عند الإعلان عن كسر يحتوي كل من البسط والمقام ثوابت رمزية، فإن ماتلاب يضع المتغير في صورة رمزية كسرية ولا يحسب قيمة الكسر كما يلي :

```
>> a=sym(2)/sym(5)
a =
2/5
```

مثال آخر يوضح طريقة التعامل مع المتغيرات الرمزية، سنحسب نتيجة جمع الكسرين ٢/٥ و ١/٣ مرة على أنهما ثوابت من النوع المضاعف ومرة على أنهما ثوابت رمزية:

```
>> a=2/5+1/3
a =
   0.7333
>> a=sym(2)/sym(5)+sym(1)/sym(3)
a =
11/15
```

في الصورة الرمزية تم جمع الكسرين بطريقة جمع الكسور الاعتيادية، حيث تم توحيد مقام كل من الكسرين وحساب البسط لكل منهما، ثم جمع البسطين فكانت النتيجة الكسر الاعتيادي ١١/١٥.

يتم الإعلان عن أي متغير في الصورة الرمزية باستخدام الأمر sym() كما رأينا، و sym هي اختصار لكلمة symbolic التي تعنى رمزي. الصورة العامة لهذا الأمر هي:

```
X=sym('X')
```

حيث تم وضع الرمز X بين العلامتين ' ' حتى يوضع هذا الرمز في المتغير X. انظر للمثال التالي:

```
a=sym('alpha')
```

حيث تم وضع الرمز 'alpha' في المتغير الرمزي a. يمكن وضع تعبير حسابي كامل مثل التعبير $\rho = \frac{1+\sqrt{5}}{2}$ كرمز في متغير رمزي كما يلي:

```
rho=sym('(1+sqrt(5))/2')
```

بعد ذلك يمكن إجراء عمليات حسابية على التغير rho كما يلي :

```
>> rho=sym('(1+sqrt(5))/2')
rho =
(1+sqrt(5))/2
>> f=rho^2-rho-1
f =
(1/2+1/2*5^(1/2))^2-3/2-1/2*5^(1/2)
>> simplify(f)
ans =
0
```

لاحظ كيف تم استخدام المتغير الرمزي rho كعنصر في تعبير كامل f وتم حساب قيمة f الرمزية ، ثم تبسيطها بالأمر simplify(f) الذي أعطى النتيجة صفراً.

انظر إلى معادلة الدرجة الثانية $f = ax^2 + bx + c$ ، هنا المتغير f يمكن وضعه في الصورة الرمزية بالأمر f=sym('a*x^2+b*x+c'). المتغيرات a و b و c و x في هذا التعبير f ليست متغيرات رمزية ؛ ولذلك لا يمكن استخدامها في عمليات حسابية رمزية مثل التفاضل والتكامل كما سنرى. لذلك في هذه الحالة لابد من تحديد هذه المتغيرات على أنها متغيرات رمزية هي الأخرى. يمكن استخدام الأمر syms a b c x الذي يضع كل هذه المتغيرات في الصورة الرمزية مرة واحدة بدلا من استخدام الأمر sym لكل متغير على حدة.

لمعرفة كيفية حل معادلتين في مجهولين باستخدام الأمر solve حاول كتابة الأوامر التالية :

```
>> syms x y
>> eq1='0.5=(200+3*x+4*y)^2/(20+2*x+3*y)^2/x'
>> eq2='10=(20+2*x+3*y)*y/x'
>> [x y]=solve(eq1,eq2,x,y)
```

بنفس الطريقة يمكن حل المعادلات الآتية، والتي سبق وأن تم حلها في التمرين المحلول ٤-٧:

$$2x-3y+4z=5$$
$$x+y+4z=10$$
$$3x+4y-2z=0$$

```
>> syms x y z;
>> eq1='2*x-3*y+4*z = 5'
>> eq2='y+4*z+x = 10'
>> eq3='-2*z+3*x+4*y = 0'
>> [x,y,z]=solve(eq1,eq2,eq3,x,y,z)
```

الأمر findsym يحدد لك أي المتغيرات في أي تعبير من النوع الرمزي.

```
>> syms a b n t x z   % هنا تم تحديد كل هذه المتغيرات من النوع الرمزي
>> f=x^n      % تحديد دالة في متغيرين
f =
x^n
>> g = sin(a*t + b)  % ودالة أخرى في ٣ متغيرات
g =
sin(a*t+b)
>> findsym(f)  % السؤال عن المتغيرات الرمزية في هذا التعبير
ans =
n, x
>> findsym(g)  % والسؤال أيضا عن المتغيرات الرمزية في هذا التعبير
ans =
a, b, t
```

الأمر subs يعوض بأي قيمة ثابتة عن أي متغير في أي تعبير:

```
>> f = 2*x^2 - 3*x + 1     % x هنا تم فرض تعبير كدالة في المتغير
f =
2*x^2-3*x+1
>> subs(f,2)  % f نريد التعويض بالقيمة ٢ عن المتغير x في التعبير
ans =
    3
>> syms x y
```

```
>> f = x^2*y + 5*x*sqrt(y) %   تعبير به المتغيرين x, y من النوع الرمزي
f =
x^2*y+5*x*y^(1/2)
>> subs(f, x, 3)  % التعويض عن المتغير x في التعبير f بالقيمة ٣
ans =
9*y+15*y^(1/2)
>> subs(f, y, 3)  % التعويض عن المتغير y في التعبير f بالقيمة ٣
ans =
3*x^2+5*x*3^(1/2)
```

يمكن تطبيق ذلك على المصفوفات، كما في المثال التالي الذي يعطي مصفوفة هلبرت:

```
>> A = hilb(3)
A =
   1.0000    0.5000    0.3333
   0.5000    0.3333    0.2500
   0.3333    0.2500    0.2000
```

بالأمر التالي سنحول هذه المصفوفة إلى الصورة الرمزية:

```
>> A = sym(A)
A =
[   1, 1/2, 1/3]
[ 1/2, 1/3, 1/4]
[ 1/3, 1/4, 1/5]
```

(٩,٢) إجراء التفاضل على المتغيرات الرمزية

يمكن إجراء التفاضل على أي متغير رمزي كما في الأمثلة التالية:

```
>> syms x
>> f = sin(5*x)  % تحديد دالة في المتغير الرمزي x
f =
sin(5*x)
>> diff(f)       % إجراء عملية التفاضل على الدالة f
ans =
5*cos(5*x)
```

مثال آخر :

```
>> g = exp(x)*cos(x)
g =
exp(x)*cos(x)
>> diff(g)
ans =
```

exp(x)*cos(x)-exp(x)*sin(x) % تذكر قانون التفاضل : الأول في تفاضل الثاني زائد الثاني في تفاضل الأول

للحصول على التفاضل الثاني للدالة g جرب الأمر التالي :

```
>> diff(g,2)
ans =
-2*exp(x)*sin(x)
```

أو الأمر التالي ، حيث ستحصل على نفس النتيجة :

```
>> diff(diff(g))
ans =
-2*exp(x)*sin(x)
```

لكي تفاضل ثابت لابد أولا من تعريف هذا الثابت في الصورة الرمزية في أي متغير كما يلي :

```
>> c = sym('5');
>> diff(c)
ans =
```

0 % الإجابة صفر ، أليس كذلك؟

ومثال آخر :

```
>> syms a b x n t theta
>> diff(x^n)
ans =
x^n*n/x
```

>> simplify(diff(x^n)) % يمكن تبسيط الإجابة السابقة باستخدام دالة التبسيط

```
ans =
x^(n-1)*n
```

بالنسبة للدوال متعددة المتغيرات، فإنه يمكن حساب التفاضل الجزئي بالنسبة لواحد فقط من متغيرات هذه الدالة كما يلي :

```
>> syms s t
>> f = sin(s*t)
f =
sin(s*t)
>> diff(f,t)   % t هنا مطلوب تفاضل f بالنسبة للمتغير فقط
ans =
cos(s*t)*s
>> diff(f,s)   % هنا مطلوب تفاضل f بالنسبة للمتغير s فقط
ans =
cos(s*t)*t
```

ويمكن إجراء الدرجات الأعلى من التفاضل الجزئي كما يلي :

```
>>diff(f,t,2)  % حيث سيحسب التفاضل الثاني للدالة f بالنسبة للمتغير t
ans =
-sin(s*t)*s^2
```

يمكن للدالة f أن تتعامل مع مصفوفات أيضا بحيث تكون كل عناصر المصفوفة من النوع الرمزي، وفي هذه الحالة سيتم التفاضل على جميع عناصر المصفوفة عنصر بعنصر كما يلي :

```
>> syms a x
>> A = [cos(a*x),sin(a*x);-sin(a*x),cos(a*x)]
A =
[  cos(a*x),  sin(a*x)]
[ -sin(a*x),  cos(a*x)]
>> diff(A)
ans =
[ -sin(a*x)*a,  cos(a*x)*a]
[ -cos(a*x)*a, -sin(a*x)*a]
```

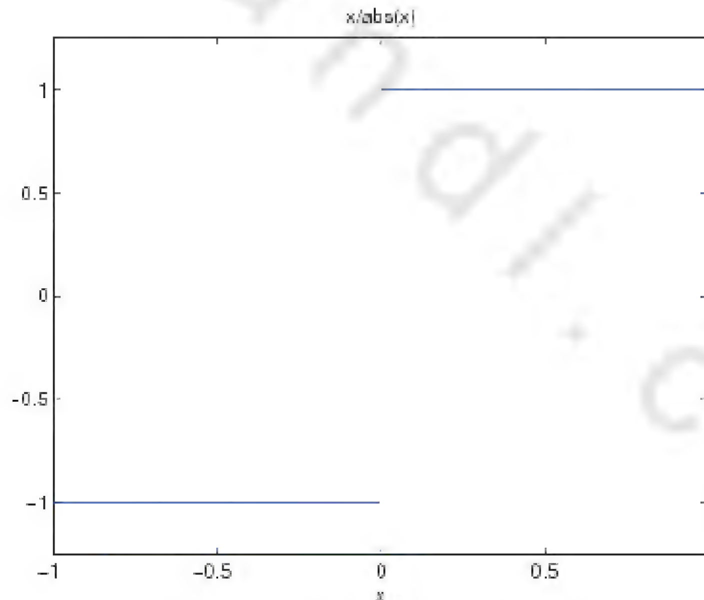
في المثال السابق كتبنا الدالة diff(A) والدالة A بها متغيران a و x، فلماذا فاضل بالنسبة ل x ولم يفاضل بالنسبة ل a ؟ في مثل هذه الأوضاع، إذا لم تحدد للماتلاب المتغير الذي سيفاضل بالنسبة له، فإنه تلقائيا يبحث عن أقرب حرف أو رمز للحرف x في الدالة ويفاضل بالنسبة له. في المثال السابق أقرب حرف للرمز x هو الحرف x نفسه ؛ ولذلك تم التفاضل بالنسبة له. القرب هنا يقصد به القرب أبجديا.

(٩,٣) النهايات Limits

يمكن حساب النهاية التي تؤول إليها أي دالة عندما يؤول أي واحد من متغيرات هذه الدالة إلى قيمة معينة. فمثلا : $\lim_{x\to 0} f(x)$ نكتب كما يلي : limit(f(x),x,0) وهذا هو الوضع التلقائي.

أي أننا إذا كتبنا limit(f(x)) فقط فإن ماتلاب سيحسب النهاية بالنسبة للمتغير x عندما يؤول إلى الصفر. بنفس الطريقة نكتب $\lim_{x\to a} f(x)$ كالتالي : lim(f(x),x,a) . انظر إلى المثال التالي :

```
>> syms h n x
>> limit( (cos(x+h) - cos(x))/h,h,0 )
ans =
-sin(x)
```



شكل (٩,١). تقارب الدالة من جهة اليمين أو اليسار.

في بعض الدوال أحادية الجانب لابد أن نحدد هل متغير الدالة يقترب من الحد المطلوب من ناحية اليمين أم من ناحية اليسار، . كمثال على ذلك انظر إلى الدالة f(x)=x/|x| الموضحة في شكل (٩,١). في هذه الدالة عندما تقترب x من الصفر قادمة من جهة اليسار، فإن الدالة f تؤول إلى ‎-١، بينما عندما تقترب x من الصفر قادمة من اليمين، فإن الدالة f تؤول إلى ‎+١ كما في الشكل. في مثل هذه الدوال، فإنه لابد من تعريف ماتلاب هل نريد حساب نهاية الدالة ونحن نقترب من جهة اليمين أم من جهة اليسار. في هذه الحالة فإن النهاية تكتب كما يلي: limit(f(x),x,0,left). انظر للمثال التالي: $\underset{x\to 0-}{Lim}\frac{x}{|x|}$ حيث يمكن حسابها بالماتلاب كما يلي:

```
>> limit(x/abs(x),x,0,'left')
ans =
-1
```

و $\underset{x\to 0+}{Lim}\frac{x}{|x|}$:

```
>> limit(x/abs(x),x,0,'right')
ans =
1
```

الدالة التي يكون لها نهاية من اليمين تختلف عن النهاية من اليسار كما في الأمثلة السابقة، فإنه لابد من تحديد اتجاه التقارب المطلوب. إذا لم يتم تحديد اتجاه التقارب، فإن ماتلاب يعطي الرسالة NaN والتي تعنى Not a Number كما يلي:

```
>> limit(x/abs(x),x,0)
ans =
NaN
```

(٩,٤) التكامل Integration

لإجراء التكامل على المتغيرات الرمزية فإن ماتلاب يستخدم الدالة int(f) للدوال في متغير واحد. يمكن تحديد متغير التكامل كما في الدالة int(f,v) التي تعطي تكامل الدالة f بالنسبة للمتغير v. إليك بعض الأمثلة على ذلك:

$\int x^n dx$ يمكن حسابها كما يلي :

```
>> syms a b x t
>> int(x^n)
ans =
x^(n+1)/(n+1)
```

$\int_0^{\pi/2} \sin(2x)dx$ والتي يمكن حسابها كما يلي :

```
>> int(sin(2*x),x,0,pi/2)
ans =
1
```

ويمكن تكامل دالة في أكثر من متغير، ولكن التكامل يكون بالنسبة لأحد هذه المتغيرات كما يلي :

```
>> g = cos(a*t + b)
g =
cos(a*t+b)
>> int(g,t)
ans =
1/a*sin(a*t+b)
```

الأوامر التالية توضح كيفية حساب نقاط الانقلاب لأي دالة، حيث يتم تفاضل الدالة ومساواة هذا التفاضل بالصفر لحساب جذور هذه المعادلة التي تمثل نقاط الانقلاب لهذه الدالة. الدالة التي سنستخدمها في هذه الخطوات هي الدالة : $f(x) = \dfrac{3x^2 + 6x - 1}{x^2 + x - 3}$

```
>> syms x
>> num = 3*x^2 + 6*x -1;
>> denom = x^2 + x - 3;
>> f = num/denom
f =
(3*x^2+6*x-1)/(x^2+x-3)
>> limit(f, inf)
ans =
3     % هذه الدالة تؤول إلى ٣ عندما تؤول x إلى مالانهاية
```

```
>> f1 = diff(f)  % تفاضل الدالة f
f1 =
(6*x+6)/(x^2+x-3)-(3*x^2+6*x-1)/(x^2+x-3)^2*(2*x+1)
>> f1 = simplify(f1)
f1 =
-(3*x^2+16*x+17)/(x^2+x-3)^2
>> pretty(f1)
            2
         3 x  + 16 x + 17
       - ----------------
            2        2
         (x  + x - 3)
```

وهي الدالة $-\frac{3x^2+16x+17}{\left(x^2+x-3\right)^2}$ التي بمساواتها بالصفر وحلها تعطي نقاط الانقلاب كما يلي :

```
>> crit_pts = solve(f1)
crit_pts =
 -8/3-1/3*13^(1/2)
 -8/3+1/3*13^(1/2)
```

وهذه النقاط هي $x1=\frac{-8+\sqrt{13}}{3}$ و $x2=\frac{-8-\sqrt{13}}{3}$ وهي نقاط الانقلاب أو النقاط الحرجة لهذه الدالة ، فهل هي نقاط انقلاب عظمى maximum أم نقاط انقلاب صغرى minimum ، حاول تحديد ذلك.

هناك أيضا التكامل المحدود الذي يتم إجراؤه بالأمر التالي :

int(f,a,b)

الذي يقوم بتكامل الدالة f من a حتى b. كمثال على ذلك التكامل التالي :

```
>> sym x;
>> f=x^7;
>> int(f,0,1)
ans =
1/8
```

وكذلك التكامل التالي :

```
>> f=log(x)*sqrt(x);
>> int(f,0,1)
ans =
-4/9
```

(٩.٥) مجموع المتواليات

تستخدم الدالة symsum لحساب مثل هذه المتواليات. مثلا من المعروف أن المجموع التالي:

$$1+\frac{1}{2^2}+\frac{1}{3^2}+\cdots$$

يساوى $\pi^2/6$ ، ويمكن حسابه باستخدام ماتلاب كما في الأوامر التالية:

```
>> sym k
>> s1 = symsum(1/k^2,1,inf)
s1 =
1/6*pi^2
```

وأيضا $1+x+x^2+x^3+\cdots$ يمكن حسابه باستخدام ماتلاب أيضا بالأوامر التالية:

```
>> syms x,k;
>> s2 = symsum(x^k,k,0,inf)
s2 =
-1/(x-1)
```

في الأمثلة السابقة inf تعني مالانهاية أو infinity. يمكن حساب متتابعة تايلور كما في المثال التالي:

```
>> syms x;
>> f = 1/(5+4*cos(x));
>> T = taylor(f,8)
T =
1/9+2/81*x^2+5/1458*x^4+49/131220*x^6
```

بخلاف كل هذه الأمثلة في المواضيع المختلفة، فإن صندوق أدوات الحسابات الرمزية Symbolic Mathematics Tool Box يحتوي على العديد من الدوال الحسابية الأخرى والطرق المختلفة المستخدمة في شتى أفرع الرياضيات، والتي يصعب ذكرها هنا بالتفصيل. على من يريد الاستزادة في هذا المجال أن يرجع إلى المساعدة التي يوفرها ماتلاب، والتي تحتوي على العديد من الأمثلة المساعدة في الكثير من التطبيقات

الرياضية. كما ذكرنا، فإن هذا الكتاب ليس كتابا تفصيليا عن ماتلاب لأن ماتلاب يدخل في جميع التخصصات الهندسية والعلمية والتي يصعب تغطيتها في أكثر من كتاب، ولكن النهج الذي نتخذه هنا هو فتح هذه المواضيع وتعريف القارئ بها وتفتيحها أمامه ونترك الباقي له لكي يستمر في الاستزادة إن أراد أو إن كان هذا الموضوع يقع في مجال تخصصه.

الفصل العاشر

أساسيات استخدام الماتلاب
في معالجة الصور الرقمية

(١٠,١) مقدمة

يوفر الماتلاب مكتبة خاصة بمعالجة الصور الرقمية تستخدم في العديد من التطبيقات، والتي تتنوع ما بين قراءة الصورة وتحسينها، أو تقسيمها وتحديد بعض الأماكن المهمة فيها، إلى أن نصل إلى استخراج معلومات واتخاذ قرار بناء على محتوى الصورة. في هذا الفصل نعرض مقدمة عن أساسيات استخدام الماتلاب في معالجة الصور الرقمية بداية من قراءة وعرض الصور، مرورا بتغيير مستويات شدة اللون وتغيير أبعاد الصورة ثم تحويلها إلى صورة ثنائية، ثم نتعرض لموضوع تحسين الصور والتنعيم باستخدام المرشحات المختلفة. وأخيرا نوضح كيفية عمل تجزئة للصورة أو تقسيم لها. يجدر بنا أن نذكر القارئ أن استخدام الماتلاب في المعالجة الرقمية للصور يجب أن تفرد له مراجع خاصة، ولكننا نضع القارئ على بداية الطريق للبحث في هذا الموضوع.

إن الصور الرقمية يمكن التعامل معها على أنها دالة ثنائية الأبعاد f(x,y) باعتبار أن x و y هما بعدان في مستوى السطح، ومقدار هذه الدالة عند أي نقطة (x,y) يعبر عن شدة إضاءة هذه النقطة أو حدتها. يطلق على هذه القيمة الكثافة intensity أو مستوى الرمادية لهذه النقطة gray level. عندما تكون الإحداثيات x و y وقيمة الدالة f(x,y) قيما محددة أو أرقاما؛ فإن الصورة في هذه الحالة يطلق عليها صورة رقمية؛ وبالتالي فإن الطرق المستخدمة لمعالجة هذه الصور ستكون أيضا طرقا رقمية نستخدم فيها الحاسبات أو المعالجات الرقمية، وهذا هو بالضبط المقصود من المعالجة الرقمية للصور.

تتكون الصورة الرقمية من عدد محدد من العناصر كل منها يشغل مكاناً معيناً وقيمة أو مستوى رماديًّا معيناً، كل من هذه العناصر يسمى عنصر صورة picture element أو pixel ونحن سنستخدم هنا نفس المنطوق، بكسل، لكثرة شيوع اللفظ. يرجع موضوع معالجة الصور إلى بداية العشرينات من القرن الماضي (عام ١٩٢٠) عندما تم نقل صور عبر المحيط من أمريكا إلى أوربا من خلال كابل بحري، وتوالى بعد ذلك تطور سريع في طرق نقل وتحسين هذه الصور، ولكن كل هذه الطرق لا يمكن أن نعتبرها طرقا رقمية؛ لأنها لم يكن الحاسب قد ظهر في هذا الوقت ولم يتم استخدامه. لذلك فإن التقدم الحقيقي في طرق المعالجة الرقمية للصور وطرق حفظها كان بعد الانفجار الذي حدث في تكنولوجيا الحاسبات التي بدأت مع نهاية السبعينات من القرن السابق.

(١٠,٢) تمثيل الصور الرقمية

افترض المصفوفة f(x,y) ثنائية الأبعاد والتي تحتوي علىM صف وN عمود؛ في هذه الحالة الإحداثيات x وy تأخذ قيما رقمية من ١ وحتى M و N على التوالي:

x = 1, 2, …, M
y = 1, 2, …, N

في هذه الحالة ؛ فإن الماتلاب يتعامل مع النقطة (١,١) على أنها مركز الصورة، ويكون الإحداثي (٢,١) يمثل البكسل المجاورة والتي تقع في الصف الأول والعمود الثاني. أما النقطة (٣,١) فهي البكسل المجاورة على نفس الصف، ... وهكذا كما في المعادلة التالية :

$$f(x,y)=\begin{bmatrix} f(1,1) & f(1,2) & \cdots & f(1,N) \\ f(2,1) & f(2,2) & \cdots & f(2,N) \\ \vdots & \vdots & \vdots & \vdots \\ f(M,1) & f(M,2) & \cdots & f(M,N) \end{bmatrix}$$

ويمكن كتابة هذه المعادلة بشكل أبسط كالتالي :

$$A=\begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,N} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,N} \\ \vdots & \vdots & \vdots & \vdots \\ a_{M,1} & a_{M,2} & \cdots & a_{M,N} \end{bmatrix}$$

يتم تمثيل الصورة في الماتلاب بأكثر من طريقة :

١-على شكل سطح ثلاثي الأبعاد ؛ المحور z يأخذ القيم المقابلة لشدة الإضاءة عند كل نقطة (x,y) وهذا موضح في شكل (١٠,١أ). هذا النوع يصبح أكثر تعقيدا وأقل فائدة كلما زاد عدد الصفوف والأعمدة في الصورة.

٢-تمثل الصورة كما هو متعارف عليه، حيث تختلف درجة اللون من بكسل لأخرى في الصورة تبعا لقيمة شدة الإضاءة أو حدتها. وهذا موضح في شكل (١٠,١ب). وكما هو واضح أن هذا التمثيل هو الأكثر شيوعا والأسهل لفهم محتوى الصورة ومعرفة مكوناتها.

٣-يتم تمثيل الصورة في شكل مصفوفة ثنائية الأبعاد كما في شكل (١٠,١ج). وهذا النوع مفيد عند عرض جزء من الصورة وتحليل القيم الرقمية عند تصميم واختبار بعض الخوارزميات.

أ

ب

```
0 0 0   0   0   0   0   0   0    ...   0   0   0   0   0   0   0   0 0
0 0 0   0   0   0   0   0   0    ...   0   0   0   0   0   0   0   0 0
0 0 0.5 0.5 0.5 0.5 0.5 0.5 0.5  ...   0.5 0.5 0.5 0.5 0.5 0.5 0.5 0 0
0 0 0.5 0.5 0.5 0.5 0.5 0.5 0.5  ...   0.5 0.5 0.5 0.5 0.5 0.5 0.5 0 0
. . .
. . .
. . .
0 0 0.5 0.5 0.5 0.5 0.5 0.5 0.5  ...   0.5 0.5 0.5 0.5 0.5 0.5 0.5 0 0
0 0 0.5 0.5 0.5 0.5 0.5 0.5 0.5  ...   0.5 0.5 0.5 0.5 0.5 0.5 0.5 0 0
0 0 0   0   0   0   0   0   0    ...   0   0   0   0   0   0   0   0 0
0 0 0   0   0   0   0   0   0    ...   0   0   0   0   0   0   0   0 0
```

ج

شكل (١٠.١). الصور المختلفة لتمثيل صورة أ- على شكل سطح ب- تبعا لمستوى الرمادي ج- على شكل مصفوفة ثنائية الأبعاد.

(١٠,٢,١) قراءة وعرض الصور الرقمية

تستخدم الدالة imread('filename') لقراءة الصور في الماتلاب ، حيث تتم كتابة اسم الملف الخاص بالصورة ويجب أن يكون الامتداد صحيحاً وكذلك يجب الأخذ في الاعتبار مكان الصورة. بمعنى أننا نستخدم اسم الصورة فقط إذا كانت الصورة موجودة في الدليل أو المجلد الحالي current directory.

```
>>imread('aseel.jpg');
```

أما إذا كانت الصورة مخزنة في مكان آخر ، فإن الرسالة التالية ستظهر :

```
Error using ==> imread at 315 ???
File "aseel.jpg" does not exist.
```

وعلى ذلك يجب كتابة المسار كاملا داخل الدالة :

```
>>imread('D:/myimages/aseel.jpg');
```

أو استخدام دالة أخرى لإضافة المسار الذي يوضح مكان الصورة ، وهي الدالة addpath كما يلي :

```
>>addpath D:/myimages
>>I=imread('aseel_01.jpg');
>>imshow(I)
```

تستخدم الدالة imshow لعرض الصورة التي تم تخزينها في المصفوفة أو المتغيرI. نلاحظ عدم وضع الفصلة المنقوطة في نهاية السطر ؛ وذلك لأنها لا تؤثر في هذه الحالة ، وعلى ذلك فإنها لا توضع عادة مع هذه الدالة. نتيجة تنفيذ هذا الأمر موضحة في شكل (١٠,٢).



شكل (١٠.٢). عرض صورة في الماتلاب.

يمكن كتابة الدالة imshow بأكثر من طريقة :

```
imshow(I,G)
```

حيث يتم عرض المصفوفة I التي تم تخزين الصورة على هيئتها باستخدام مستويات لشدة اللون عددها G ، أما عند عدم ذكر القيمة فإن الماتلاب يستخدم القيمة ٢٥٦ كقيمة افتراضية default .

أو يتم تحديد القيمتين العظمى high والصغرى low المستخدمتين لشدة الإضاءة لكل بكسل في الصورة في الدالة نفسها. وعند عرض الصورة، فإن القيم الأقل من القيمة low تأخذ اللون الأسود، والقيم الأعلى من القيمة high تأخذ اللون الأبيض. وتتم في هذه الحالة كتابة الدالة بالشكل التالي :

```
imshow(I,[low high])
```

في بعض التطبيقات التي يكون فيها dynamic range (نطاق أو مدى مستويات الرمادي الذي تحتاجه الصورة) صغيراً أي أن مدى التغير بين قيم البكسلات الموجودة في الصورة تغير بسيط والبكسلات تأخذ قيما متقاربة، أو بعض القيم موجبة وبعضها سالبة. في هذه الحالة من الأفضل أن يتم عرض كل صورة باستخدام عدد من مستويات اللون تبعا لهذه الصورة. وعلى ذلك تكون القيمة high هي أكبر قيمة لشدة الإضاءة والقيمة low هي أقل قيمة لشدة الإضاءة في الصورة. تتم كتابة الدالة في هذه الحالة بالشكل التالي :

```
imshow(I,[ ])
```

في كل مرة نستخدم هذه الدالة نلاحظ أن الصورة المعروضة الحالية تتغير. في حالة الرغبة في عرض أكثر من صورة نستخدم الأمر figure مع كل أمر لعرض الصورة. في بعض الحالات التي نفضل وجود أكثر من صورة في نفس الشكل يمكن أن نستخدم الدالة subplot التي سبق استخدامها مع دوال الرسم. لاحظ الفرق بين الصور المعروضة في شكل (١٠،٣) والذي يوضح تأثير المتغير G كما في الخطوات التالية :

```
>> figure,
>> subplot(1,3,1), imshow(I,[]), title('G = [ ]')
>> subplot(1,3,2), imshow(I,[ 64 128]), title('G = [64 128]')
>> subplot(1,3,3), imshow(I,[ 0 100]), title('G = [0 100]')
```



شكل (١٠،٣). تأثير تغيير مستويات شدة اللون على الصورة أ– القيمة محددة تبعا للصورة نفسها ب– القيمة بين ٦٤ و ١٢٨ ج– القيمة بين ٠ و ٢٠٠.

من الممكن استخدام الدالة impixelinfo لمعرفة شدة إضاءة أي بكسل في الصورة بطريقة تفاعلية. فعند استخدام هذه الدالة يجب أولا أن نعرض الصورة ثم نكتب الدالة فنجد مؤشرا يتحرك مع حركة الفأرة فوق الصورة، وعند كل بكسل يعرض لنا إحداثياتها وشدة الإضاءة. الدالة imdistline تستخدم لحساب المسافة بين أي نقطتين على الصورة.

لمعرفة حجم الصورة نستخدم الدالة size والتي تعطي أبعاد الصورة (الصفوف والأعمدة):

```
>>size(I)
ans =
1145    825
```

هذا يعني أن الصورة عبارة عن ١١٤٥ صف و ٨٢٥ عمود، في بعض الصور يكون خرج هذه الدالة عبارة عن ثلاث قيم مثل: ٣ ٨٢٥ ١١٤٥ في هذه الحالة الرقمان الأول والثاني يمثلان عدد الصفوف والأعمدة على الترتيب أما الرقم ٣ فيعني أن كل بكسل لها ٣ قيم مختلفة لدرجة شدة الإضاءة في تمثل الأبعاد الثلاثة للصورة الملونة (أحمر- أخضر- أزرق) على التوالي، أو كما هو شائع (RGB).

(١٠,٢,٢) كتابة الصور الرقمية

في كثير من التطبيقات، وخصوصا الطبية، تكون هناك حاجة لحفظ الصور الخاصة بالمرضى و كتابة بعض التعليقات الخاصة بهم. يمكن كتابة الصور على الديسك، أي يتم حفظها لحين الحاجة إليها باستخدام الدالة imwrite .

```
>>imwrite (I,'aseel_01.jpg');
```

يقوم هذا الأمر بكتابة أو تخزين الصورة الممثلة في المصفوفة I بالاسم aseel_01 وله الامتداد jpg. ، يجب الأخذ في الاعتبار أن يأخذ اسم الملف أحد الامتدادات الآتية : tif, tiff, jpg, jpeg, bmp, png, xwd وعند كتابة الأمر بالشكل السابق يتم تخزين الصورة في الدليل الحالي أما إذا أردنا وضعها في مكان آخر، فيجب كتابة المسار كاملا. يمكن التحكم في كثير من خصائص الصورة عند كتابتها باستخدام هذه الدالة عن طريق تحديد بعض المتغيرات مثل: الجودة quality والضغط compression والمقدرة التحليلية resolution و متغيرات أخرى عديدة. وفي هذه الحالة يكون الشكل العام لاستخدام الدالة هو :

```
imwrite(I, 'filename', 'param1',val1, 'param2',val2,...)
```

يمكن التعرف على كل هذه المتغيرات عن طريق الأمر :

```
>> help imwrite
```

ولنترك للقارئ استعمال متغيرات مختلفة وملاحظة التغير في الصورة في كل مرة.

كذلك استخدام الدالة imfinfo لعرض المعلومات الخاصة بكل صورة. مثل الأمر التالي :

```
>> imwrite(I,'aseel_01.jpg')
>> imfinfo aseel_01.jpg
= ans
Filename: 'aseel_01.jpg'
FileModDate: '11-Jun-2011 23:46:02'
FileSize: 435256
Format: 'jpg'
FormatVersion : ''
Width: 1944
Height: 2592
BitDepth: 8
ColorType: 'grayscale'
FormatSignature: ''
NumberOfSamples: 1
CodingMethod: 'Huffman'
CodingProcess: 'Sequential'
Comment: {}
```

(١٠,٢,٣) أنواع الصور

يتعامل الماتلاب مع أربعة أنواع من الصور:

١– صورة الكثافة intensity image

أو صورة متدرجة الرمادي grayscale image. في هذا النوع من الصور فإن البكسلات تأخذ القيم من ٠ إلى ٢٥٥ إذا كانت البيانات من النوع uint8 حيث يمثل الصفر اللون الأسود و٢٥٥ اللون الأبيض وما بينهما تدرجات اللون الرمادي. أما في حالة uint16 تكون القيم من ٠ إلى ٦٥٥٣٥.

٢– صورة الفهرسة indexed image

في هذا النوع من الصور، فإن البكسلات تأخذ القيم من ٠ إلى ١ وتكون البيانات من النوع doubl.

٣– صورة ثنائية binary image

في هذا النوع من الصور، فإن البكسلات تأخذ قيمتين فقط ٠ للون الأسود أو ١ للون الأبيض وتكون البيانات من النوع logical.

٤– صورة ملونة RGB image

في هذا النوع من الصور، فإن كل بكسل تأخذ ثلاث قيم مختلفة تمثل الألوان الأحمر، الأخضر، والأزرق. وتكون البيانات من النوع uint8.

يمكن التحويل بين هذه الأنواع المختلفة للصور باستخدام الدوال:

.im2bw, im2double, mat2gray, im2uint16, im2uint8

جدول (١٠.١) يلخص هذه الدوال و يوضح نوع البيانات في الدخل والخرج :

جدول (١٠.١). دوال التحويل بين أنواع مختلفة من الصور.

الدالة	نوع البيانات الممكن للصورة في الدخل	مدى قيم شدة الإضاءة في صورة الخرج	نوع البيانات في الصورة التي تم التحويل لها
im2uint8	logical, uint16, double	[0 255]	uint8
im2uint16	logical, uint8, double	[0 65535]	uint16
mat2gray	logical, uint8, uint16	[0 1]	double
im2double	logical, uint8, uint16	[0 1]	double
im2bw	uint8, uint16, double	0 or 1	logical

(١٠.٢.٤) المقدرة التحليلية

من أهم خواص الصورة التي تتأثر بها العين مباشرة خاصية التحديد أو المقدرة التحليلية resolution والتي على ضوئها تتحدد جودة جهاز أو طريقة عرض الصورة. كما ذكرنا سابقا، فإن الصورة الرقمية يتم عرضها في صورة عدد من البكسلات في وحدة المساحة. فكلما زاد عدد البكسلات في وحدة المساحة زادت جودة الصورة، فشاشة الصورة المقسمة إلى ٥١٢×٥١٢ بكسل بالطبع ستكون أفضل بكثير من شاشة نفس المساحة ولكنها مقسمة إلى ١٦×١٦ بكسل. تخيل أنك تريد عمل صورة مكبرة بطريقة يدوية عن طريق تقسيم هذه الصورة الأصلية إلى مربعات، ثم نقل محتويات كل مربع على حده من الصورة الأصلية إلى الصورة المكبرة. بالطبع، فإن جودة عملية النسخ ستكون أفضل بكثير كلما كان عدد المربعات المستخدمة في عملية النسخ أكبر، هنا عدد المربعات المستخدمة يقابل عدد البكسلات أو عناصر الصورة التي ذكرناها. شكل (١٠.٤) يبين صورة 'منى' وقد تم تصغيرها على مراحل عن طريق حذف صف من البكسلات من بين كل صفين، وأيضا عمود من البكسلات من بين كل عمودين حيث يتم تقسيم الصورة إلى عدد من الصفوف وعدد من الأعمدة من البكسلات.

الدالة imresize تستخدم لتعديل أبعاد الصورة إلى عدد معين من الصفوف والأعمدة وشكلها العام:

```
B = IMRESIZE(A, SCALE)
B = IMRESIZE(A, [NUMROWS NUMCOLS])
```

شكل (١٠.٤أ) تم الحصول عليه بتنفيذ الأوامر التالية:

```
>> M = imread('mona001.jpg');
>> M1 = imresize(M,[256 256]);
>> figure, imshow(M1)
```

الصورة المعروضة في شكل (١٠.٤أ) تتكون من ٢٥٦ صف و ٢٥٦ عمود من البكسلات؛ ولذلك فإننا نقول أنها ٢٥٦×٢٥٦ بكسل، وتم الحصول عليها بتغيير عدد الصفوف والأعمدة باستخدام الدالة imresize. شكل (١٠.٤ب) يوضح الصورة وقد أصبحت ١٢٨×١٢٨ بكسل وذلك بتغيير عدد الصفوف وعدد الأعمدة في الدالة imresize كما في الأمر التالي.

```
>> M2 = imresize(M,[128 128]);
```

الصور المعروضة في شكلي (١٠.٤ج و ١٠.٤د) نتيجة تغيير عدد الصفوف والأعمدة لتكون الصور ٦٤×٦٤ و ٣٢×٣٢ على التوالي.



شكل (١٠.٤). أ- الصورة الأصلية ٢٥٦×٢٥٦ بكسل ب- نفس الصورة ١٢٨×١٢٨ بكسل ج- نفس الصورة ٦٤×٦٤بكسل د- نفس الصورة ٣٢×٣٢ بكسل.

لبيان تأثير تغيير عدد الصفوف والأعمدة على شكل الصورة؛ نعرض الصور السابقة في نفس المساحة كما في شكل (١٠،٥) باستخدام الخطوات التالية:

```
>> figure,
>> M1 = imresize(M,[256 256]); subplot (2,2,1), imshow(M1)
>> M2 = imresize(M,[128 128]); subplot (2,2,2), imshow(M2)
>> M3 = imresize(M,[64 64]); subplot (2,2,3), imshow(M3)
>> M4 = imresize(M,[32 32]);  subplot (2,2,4), imshow(M4)
```

شكل (١٠،٥) يبين تأثير جودة التحديد resolution حيث نلاحظ في شكل (١٠،٥أ) يمثل الصورة الأصلية ٢٥٦×٢٥٦ بكسل، وشكل (١٠،٥ب) يمثل الصورة ١٢٨×١٢٨ بكسل معروضة في نفس مساحة الصورة الأصلية، وشكل (١٠،٥ج) عبارة عن صورة ٦٤×٦٤ بكسل معروضة في نفس مساحة الصورة الأصلية، وأخيرا شكل (١٠،٥د) يمثل صورة ٣٢×٣٢ بكسل معروضة في نفس مساحة الصورة الأصلية. لاحظ في هذه الأشكال ظهور عيب الصندقة blocking effect حيث تظهر الصورة في صورة بلوكات تشابه بلوكات لوحة الشطرنج تتسبب في عدم وضوح الصورة وعدم تحديدها، وهذا عيب معروف في وسط المعالجة الرقمية للصور، وهناك العديد من طرق التخلص منه.



شكل (١٠،٥). أ- الصورة الأصلية ب- صورة ١٢٨×١٢٨بكسل ج- صورة ٦٤×٦٤ د- صورة ٣٢×٣٢ بنفس مساحة الصورة الأصلية.

كما ذكرنا فإن الصورة الرقمية عبارة عن مصفوفة من الأرقام؛ كل رقم يمثل كثافة intensity أو شدة الإضاءة عند هذه البكسل. الرقم الممثل لكثافة كل نقطة يتم التعبير عنه بعدد من البتات في حالة الصور الرمادية. فمثلا في حالة استخدام بايت كاملة (٨ بت) لتمثيل هذه الأرقام، فإن كل رقم من هذه الأرقام سيتراوح من صفر حتى ٢٥٥، أي أن عدد مستويات الرمادية في هذه الحالة سيكون ٢٥٦ مستوى. بفرض أن لدينا صورة مساحتها ٦٤×٦٤ بكسل فإن هذه الصورة ستحتاج لذاكرة مقدارها

٦٤×٦٤×١ بايت =٤٠٩٦ بايت. بالنسبة للصور الملونة RGB فإن كل بكسل تمثل بثلاثة بايت حيث تمثل درجة كل لون ببايت، بايت تمثل كثافة اللون الأحمر، وأخرى تمثل كثافة اللون الأخضر، والثالثة تمثل كثافة اللون الأزرق، بحيث عند جمع الثلاثة مركبات نحصل على اللون الطبيعي للبكسل. في هذه الحالة فإن الصورة السابقة ستحتاج لذاكرة أكبر حيث ستكون ٦٤×٦٤×٣ بكسل = ١٢٢٨٨ بكسل .

يتضح مما سبق أنه بزيادة عدد البكسلات في وحدة المساحة تكون الصورة أفضل أو بمعنى آخر التفاصيل الدقيقة في الصورة تصبح أكثر وضوحا. ومن العوامل المؤثرة في وضوح الصورة أيضا اختيار عدد مستويات تمثيل مستوى الرمادية في الصورة. فمثلا عند تمثيل الصورة بمستويين رماديين فقط فهذا يعني أن البكسلات إما تأخذ اللون الأبيض وأما الأسود وهذا يؤدى إلى أننا نحتاج ١ بت لتمثيل كثافة اللون عند أي بكسل (لأن القيمة في هذه الحالة صفر أو واحد). هذه تمثل الصور الثنائية التي ذكرناها كأحد أنواع الصور التي يتعامل معها الماتلاب. يمكن تحويل الصور التي تحتوي على عدد أكبر من مستويات الرمادي عن طريق وضع مستوى تشبع معين threshold، بحيث يتم وضع جميع البكسلات التي لها مستوى رمادي أكبر من أو يساوي مستوى التشبع بالقيمة واحد. ووضع جميع البكسلات التي لها مستوى رمادي أقل من هذا المستوى بالقيمة صفر. من الجدير بالذكر أن اختيار القيمة المناسبة لحد التشبع ليست بالأمر اليسير، وهي من المشكلات الشائعة في كثير من التطبيقات، حيث تختلف من صورة لأخرى ومن تطبيق لآخر. يؤثر اختيار حد التشبع على شكل الصورة الناتجة. يمكن استخدام الدالة imshow أو الدالة im2bw للحصول على صورة ثنائية. حاول تجربة الأمرين التاليين ولاحظ الفرق بين الصورتين الناتجتين.

```
>>figure, imshow(I>128)
>>J = im2bw(I, 0.5);
>>figure, imshow (J)
```

شكل (١٠،٦) يبين نفس صورة أسيل في المستويين الأبيض والأسود باستخدام أكثر من مستوى فصل وباستخدام الدالة im2bw وصورتها العامة كالآتي :

BW =im2bw(I,level)

حيث BW هي الصورة الأبيض والأسود الناتجة، و I هي الصورة المراد تمثيلها بالمستويين الأبيض والأسود و level هو المستوى الذي ستكون كثافة البكسلات الأعلى منه باللون الأبيض والبكسلات التي ستكون كثافتها أقل منه باللون الأسود كما في شكل (١٠،٦).

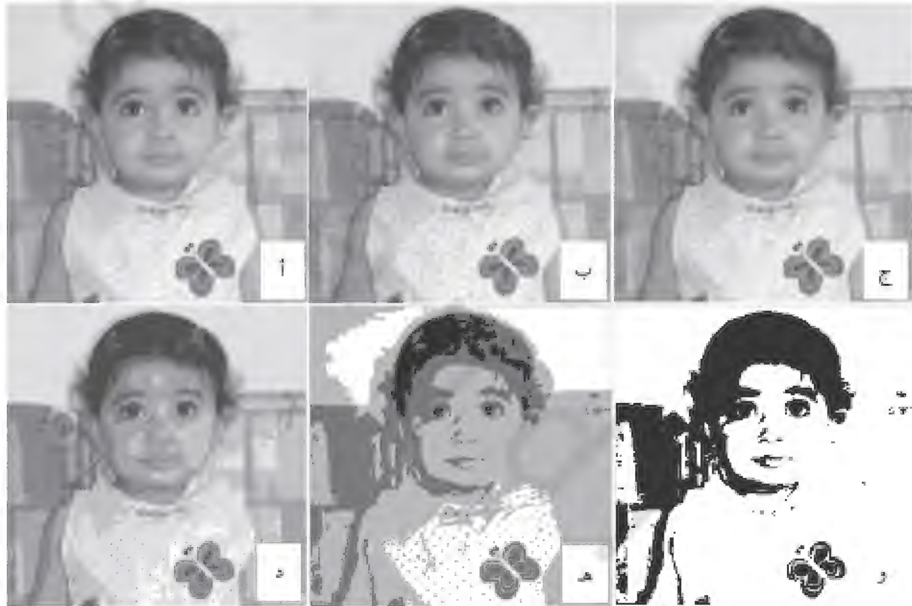


شكل (١٠،٦). الحصول على صورة أبيض وأسود من الصورة الرمادية بأكثر من مستوى تشبع أ- الصورة الأصلية ب- level=0.4 ج- level=0.5 د- level=0.6 ه- level=0.65 و- level=0.7.

شكل (١٠،٧) يبين تمثيل الكثافة بأعداد مختلفة من البتات بدءا من ٧ بت = ١٢٨ مستوى وانتهاء بـ ١ بت = مستويين فقط. نلاحظ أنه بزيادة عدد مستويات تمثيل مستوى الرمادية، فإن الصورة تكون أفضل ومريحة أكثر للعين البشرية. الدالة histeq تستخدم

لعرض صورة بعدد معين من المستويات، وسيتم شرحها لاحقا في هذا الفصل. ولكن نترك المستخدم ليجرب الأوامر التالية للحصول على الصور الموجودة في شكل (١٠,٧)

```
>>figure, subplot(2,3,1), histeq(I,128)
>> subplot(2,3,2), histeq(I,64)
>> subplot(2,3,3), histeq(I,32)
>> subplot(2,3,4), histeq(I,16)
>> subplot(2,3,5), histeq(I,8)
>> subplot(2,3,6), histeq(I,4)
```



شكل (١٠,٧). تمثيل الصورة بمستويات رمادية مختلفة أ- ١٢٨ مستوى ب- ٣٢ مستوى ج- ١٦ مستوى د- ٨ مستويات هـ-٤ مستويات و- مستويان.

(١٠,٣) تحسين الصور Image Enhancement

المقصود بتحسين الصورة هو الحصول على صورة أفضل من الصورة الأصلية باستخدام طريقة أو خوارزم معين. خوارزمات تحسين الصورة كثيرة ومتعددة وتعتمد بدرجة كبيرة على الصورة نفسها، فالطريقة التي تعطي نتائج جيدة مع صور

الكاميرات الضوئية ليس بالضرورة أن تعطي نفس الجودة مع صور أشعة إكس. تنقسم خوارزمات تحسين الصورة إلى قسمين، قسم يعمل في النطاق المكاني أو المساحي spatial domain والذي يعمل على مساحة الصورة وعلى بكسالاتها، وأما القسم الثاني فيعمل على الصورة وهي في النطاق الترددي frequency domain أي بعد أخذ محول فورير لها.

(١٠،٤) طرق التحسين في نطاق مساحة الصورة

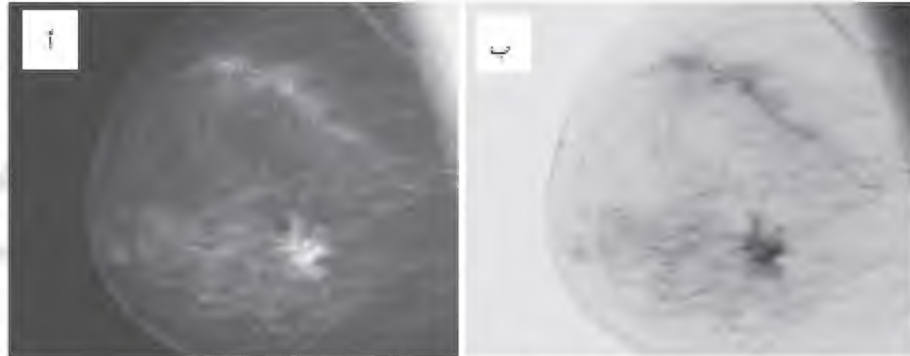
يتم تحويل مستوى الرمادي أو شدة إضاءة بكسل من قيمة إلى أخرى بهدف تحسين الصورة بإحدى الطرق الآتية :

استخدام دوال تحويل

مثل الدالة imadjust والتي تغير كل بكسل إلى قيمة مقابلة يتم تحديدها عن طريق بعض المتغيرات. الصورة العامة لهذه الدالة :

```
T = imadjust(I,[low_in  high_in], [low_out  high_out], gamma)
```

لاحظ أنه عند كتابة الأمر بالشكل T = imadjust(I,[0 1], [1 0]) فإنه يحول الصورة إلى الخرج الموضح في شكل (١٠،٨) والذي يكافئ الصورة السالبة حيث استبدلنا القيمة العظمى بالصغرى والقيمة الصغرى بالعظمى. يمكن أن نقول إن الأمر بهذا الشكل يكافئ استخدام الدالة imcomplement. في العادة يكون هناك صور إذا تم عكسها تكون الصورة أفضل ومبينة للكثير من التفاصيل كما في شكل (١٠،٨) الذي يبين صورة أشعة إكس لثدي يتم فحصه لتشخيص وجود سرطان من عدمه في هذا الثدي، حيث نلاحظ أن الصورة الأصلية تحتوي الكثير من السواد مما أخفى الكثير من تفاصيلها، ولكن بعكسها ظهر الكثير من التفاصيل.



شكل (٨.١٠). ماموجرام أ- الصورة الأصلية ب- الصورة السالبة.

المتغير gamma يحدد طريقة الإسقاط أو التحويل بين قيم البكسلات في الصورة الأصلية إلى الصورة في الخرج (هل العلاقة خطية أم غير خطية؟). عند قيمة gamma تساوي ١ العلاقة تكون خطية. أما عند قيمة gamma أقل من ١ فإن صورة الخرج تكون أفتح (أكثر إضاءة) من الصورة الأصلية، وعندما تكون قيمة gamma أكبر من ١ تكون الصورة الناتجة أغمق من الصورة الأصلية. لتوضيح تأثير هذا المتغير فإننا ننصح القارئ بكتابة الأوامر وملاحظة الصورة الناتجة عند قيم مختلفة للمتغيرgamma.

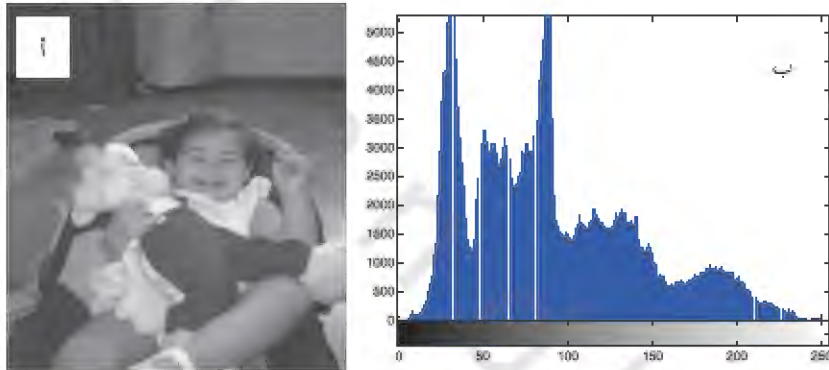
```
>> N=imread('tire.tif');
>> T1=imadjust(N,[],[],1); figure,imshow(T1);
>> T2=imadjust(N,[],[],0.4); figure,imshow(T2);
>> T3=imadjust(N,[],[],3); figure,imshow(T3);
```

استخدام المدرج الإحصائي

الهستوجرام Histogram أو المدرج الإحصائي هو أحد الطرق الشهيرة التي تعمل على بكسلات الصورة، أي في النطاق المساحي للصورة. بفرض أن لدينا صورة رقمية عدد مستوياتها الرمادية يمتد من المستوى صفر إلى المستوى L-1. هستوجرام هذه الصورة يمكن كتابته $h(r_k)=n_k$ حيث $k=0,1,2,...,L-1$ و r_k هو المستوى الرمادي رقم k، و n_k هو عدد البكسلات التي لها هذا المستوى الرمادي. فمثلا في صورة لها ٢٥٦

مستوى رمادي، الهستوجرام يحسب عدد البكسلات في المستوى صفر، وعدد البكسلات في المستوى الأول و الثاني و ... وهكذا إلى المستوى ٢٥٥. يتم تمثيل الهستوجرام على محورين، الأفقي يمثل المستويات من صفر حتى ٢٥٥، والرأسي أعمدة كل عمود فيها يمثل عدد البكسلات في هذا المستوى. شكل (١٠.٩) يبين صورة سهيلة وبجوارها الهستوجرام الخاص بها نتيجة تنفيذ الأوامر التالية :

```
>> I = imread('sohayla.jpg');
>> figure, imshow(I)
>> figure, imhist(I)
```



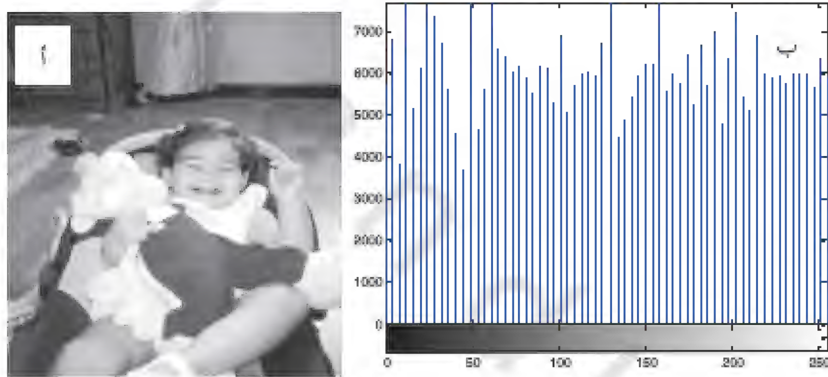
شكل (١٠-٩) أ- الصورة الأصلية ب- الهستوجرام.

لاحظ الشريط الموجود في أسفل الهستوجرام، والذي يبين أن أقصى مستوى في اللون الأبيض هو المستوى ٢٥٦ بينما المستوى صفر يمثل أقصى اللون الأسود. نلاحظ أيضا من الهستوجرام أن الصورة تميل إلى اللون الغامق، حيث نلاحظ تمركز الهستوجرام ناحية اللون الأسود. المفروض حتى تكون الصورة ذات تباين جيد أن يكون الهستوجرام الخاص بها منتظم تقريبا على كل المستويات الرمادية أو يأخذ شكل الجرس بحيث يكون عاليا قليلا عند المستويات الرمادية المتوسطة ويقل تدريجيا عند مستويات الأطراف. هناك طرق لتحسين الصورة عن طريق ضبط الهستوجرام يضيق المكان لشرحها هنا، ولكن يمكن قراءة تفاصيلها في أي مرجع من مراجع معالجة

الصور. برنامج الماتلاب به دالة تقوم بضبط هستوجرام أي صورة تسمى histogram equalization وتكتب كما يلي:

J = histeq(I, n)

حيث J هي الصورة الناتجة بعد تعديل هستوجرام الصورة I و n تمثل عدد المستويات التي يتم مستوى الرمادية بها، والتي وضحنا تأثيرها على الصورة في شكل (١٠،٧). باستخدام الدالة السابقة لتعديل هستوجرام صورة سهيلة في شكل (١٠،٩أ) نحصل على الصورة الجديدة والهستوجرام الخاص بها كما في شكل (١٠،١٠).



شكل (١٠،١٠) أ- الصورة بعد تعديلها عن طريق تعديل الهستوجرام ب- الهستوجرام الجديد.

لاحظ التباين الزائد في الصورة الناتجة، حيث اللون الأبيض أصبح أكثر بياضا والأسود أصبح أكثر سوادا. ربما لو وضعنا الصورتين متجاورتين يمكننا أن نلاحظ هذا الفرق بسهولة. إن تعديل الصورة عن طريق تعديل الهستوجرام يعتمد بدرجة كبيرة على خبرة المستخدم وعلى عين الرائي.

يمكن أيضا عن طريق الهستوجرام أن نغير في شدة إضاءة البكسلات المختلفة عن طريق ما يسمى histogram specification. أي نحول الهستوجرام من شكل إلى آخر مما ينتج عنه تغيير في الصورة. تستخدم هذه الطريقة في بعض التطبيقات الخاصة، والتي

تحتاج أن نتعامل مع بعض مستويات الإضاءة داخل الصورة بطريقة معينة ومع بعض المستويات الأخرى بطريقة أخرى.

استخدام المرشحات أو الفلاتر في النطاق المساحي للصورة

المرشح الذي نعنيه هنا هو مرشح يعمل على بكسلات الصورة أو يعمل على الصورة وهي في النطاق المساحي. مثال ذلك أن نمثل مستوى الرمادية لكل بكسل بمتوسط مستوى الإضاءة لعدد من البكسلات المحيطة بها والموجودة داخل نافذة معينة. من أمثلة ذلك :

١- **مرشحات تنعيم الصورة:** تعتمد في نظريتها على استبدال مستوى الرمادية لكل بكسل بمتوسط مستوى الرمادية للبكسلات المحيطة بها على حسب مساحة المرشح. برنامج الماتلاب لديه الدالة imfilter التي تقوم بترشيح الصورة المعطاه تبعا لنوع المرشح المستخدم أيضا. الدالة في صورتها العامة هي :

imfilter(input_image, mask, filtering_mode, boundary_option, size_option)

الخطوات التالية تبين مثال على ذلك :

```
>>I=imread('d:\aseel\aseel256.jpg');
>>J = rgb2gray(I);
>>J = imresize(J,[256 256]);
>>imshow(J); title('Original Image')
>>h = ones(11,11) / 121;
>>I2 = imfilter(J,h,'conv');
>>figure; imshow(I2); title('Filtered Image);
```

حيث تم تحديد المرشح الذي سيتم استخدامه بالمصفوفة h وبعد ذلك تم إجراء عرض للصورتين الأصلية والمرشحة كما في شكل (١٠،١١). الدالة imfilter يمكن استخدامها بدون الاختيار 'conv' حيث في هذه الحالة سيتم ضرب بكسلات المرشح مباشرة في البكسلات المقابلة لها في الصورة الأصلية بدون عملية الدوران ١٨٠ درجة كما في حالة دالة الالتفاف convolution. هذه هي الحالة التلقائية للدالة imfilter ، أي أنه إذا لم تكتب بها الاختيار 'conv' فإن الدالة لن تنفذ الضرب الالتفافي ، ولكنها ستنفذ

الضرب العلاقي correlation فحاول تنفيذ نفس البرنامج السابق بنفس حجم النافذة أو المرشح (١١×١١ بكسل وكلها وحايد) وانظر هل هناك فرق بين الحالتين أم لا؟ المفروض ألا يكون هناك فرق؛ لأن مصفوفة المرشح أو نافذة الترشيح كلها وحايد ولن تتأثر بعملية الدوران ١٨٠ درجة نتيجة إجراء الضرب الالتفافي.



شكل (١٠،١١). أ– الصورة الأصلية ب– نفس الصورة بعد تطبيق مرشح التنعيم.

هناك ملاحظة ثانية على الصورة الناتجة من مرشح التنعيم في الشكل (١٠،١١) وهي وجود إطار أسود أو حدود حول الصورة الناتجة من الترشيح، وهذا الإطار أو هذا الحد لم يكن موجودا في الصورة الأصلية، فما هو سبب وجود هذا الإطار؟ سبب وجود هذا الإطار نفهمه لو فهمنا طريقة تطبيق هذا المرشح. كما ذكرنا أنه يتم تطبيق مساحة المرشح على مساحة الصورة لتعديل أو تنعيم مستوى نقطة المركز في مساحة المرشح، ثم نبدأ في تحريك المرشح بالنسبة للصورة من اليسار لليمين ومن أعلى لأسفل إلى أن ننتهي من مسح الصورة كلها.

السؤال الآن هو، ما هو موقف النقط الموجودة على حافة الصورة أو على إطارها؟ عندما تكون واحدة من هذه النقط هي مركز مساحة المرشح فإن جزء من مساحة المرشح سيقع خارج نقاط الصورة حيث لا توجد بكسلات معروفة للصورة يمكن الضرب فيها، فما هو الموقف في هذه الحالة؟ وكيف سنتعامل مع بكسلات المرشح التي تقع خارج نطاق الصورة؟ هناك أكثر من حل لهذا الموقف، أحدها: أن يتم فرض مستوى رمادية هذه البكسلات بأصفار وهذا ما يفعله الماتلاب فعلا وهو السبب في ظهور الإطار الأسود حول الصورة وهو ما يسمى zero padding. هناك حل آخر وهو أن كل بكسل خارج نطاق الصورة تأخذ نفس مستوى الرمادية في النقطة المجاورة لها على حافة الصورة أو بمعنى آخر يتم تكرار نقاط الحافة كبكسلات خارج الصورة وهو ما يسمى border replication. ويمكن عمل ذلك في الدالة imfilter بإضافة كلمة replicate لها كما يلي، وكما هو موضح في شكل (١٠،١٢) حيث نلاحظ اختفاء الحافة الخارجية تقريبا بالمقارنة بالصورة الموجودة في شكل (١٠،١١ب).



شكل (١٠،١٢). تأثير المعاملة مع حواف الصورة.

٢- مرشحات الوسط: هذا النوع من المرشحات مفيد جدا في إزالة نوع معين من الضوضاء التي تقع على الصورة والتي تكون في صورة نقاط سوداء جدا في وسط أقل سوادا، أو العكس نقاط بيضاء جدا في وسط أقل بياضا أيضا. هذا النوع من الضوضاء يسمى ضوضاء الملح والفلفل salt and pepper noise لأنها تشبه حبات الفلفل الأسود المنتشرة في الملح الأبيض. مرشح الوسط median filter عبارة عن نافذة مربعة لا تحتوي أي معاملات تمسح بها الصورة كما سبق، ولكننا نقرأ مستويات بكسلات الصورة تحت هذه النافذة ونرتبها ترتيبا تصاعديا ونأخذ القيمة المتوسطة فيها ونبدل بها قيمة البكسل التي نرشحها أو نعالجها، والتي هي البكسل الموجودة في مركز النافذة. فمثلا لو فرضنا أن بكسلات الصورة التي كانت تحت نافذة أبعادها ٣×٣ مستوياتها الرمادية كالتالي (٥٠، ١٠، ٨٠، ٣٠، ٠، ١٠٠، ٢٥، ٨٨، ٩٠). لتطبيق مرشح الوسط على هذه النافذة نقوم بترتيب البكسلات السابقة ترتيبا تصاعديا من اليسار لليمين كالتالي (٠، ١٠، ٢٥، ٣٠، ٥٠، ٨٠، ٨٨، ٩٠، ١٠٠)، وعلى ذلك تكون القيمة الوسطى هي القيمة ٥٠ حيث إن قبلها ٤ قراءات أقل منها وبعدها ٤ قراءات أكبر منها كما ترى. بعد ذلك نقوم باستبدال قيمة البكسل التي في مركز المرشح بالقيمة ٥٠. لاحظ أننا نبحث عن القيمة الوسط في النافذة، أي التي في منتصف القائمة بعد الترتيب وقبلها عدد من النقاط يساوي العدد الذي بعدها. البعض يعتقد أننا نحسب متوسط نقاط النافذة وهذا خطأ. برنامج الماتلاب يوفر الكثير من الدوال للمساعدة في ذلك ومنها مثلا الدالة:

```
J = imnoise(J,'salt & pepper',0.02);
```

التي تضيف الضوضاء من نوع الملح والفلفل للصورة بنسبة ٠,٠٢ من عدد بكسلات الصورة، وكذلك فإن كثافة كل بكسل أو مستوى الرمادية لها يكون ٠,٠٢ أيضا. البرنامج التالي سيضيف هذا النوع من الضوضاء على صورة أسيل الأصلية ثم

يستخدم مرشح وسط بالأبعاد ٣×٣ لإزالة هذه الضوضاء كما في شكل (١٠،١٣) الذي يبين الصورة الأصلية والصورة مع الضوضاء ثم الصورة بعد إزالة هذه الضوضاء.



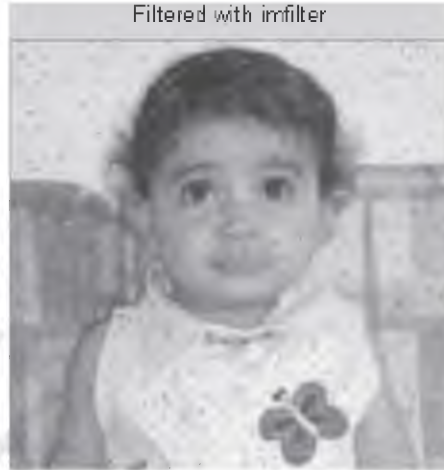
شكل (١٠،١٣). تأثير مرشح الوسط median filter على ضوضاء الملح والفلفل أ- الصورة الأصلية ب- الصورة مضافا إليها الضوضاء ج- الصورة بعد إزالة الضوضاء.

```
I=imread('d:\aseel\aseel256.jpg');
J = rgb2gray(I);
J = imresize(J,[256 256]);
imshow(J);title('Original Image')
J = imnoise(J,'salt & pepper',0.02);
figure, imshow(J)
L = medfilt2(J,[3 3]);
figure, imshow(L)
```

لنرى مدى جودة مرشح الوسط في إزالة هذا النوع من الضوضاء سنحاول إزالة هذه الضوضاء بمرشح تنعيم كالتالي:

```
h = ones(3,3) / 9;
I2 = imfilter(J,h);
```

شكل (١٠،١٤) يبين كيف أن مرشح التنعيم لم يُزِلْ هذه الضوضاء كما فعل مرشح الوسط.



شكل (١٠,١٤). إزالة ضوضاء الملح والفلفل باستخدام مرشح تنعيم.

٣- **مرشحات الإظهار**: إظهار الصورة نقصد به التخلص مما بها من ضبابية إن وجد، وكذلك التأكيد أو إظهار تفاصيل الصورة بما فيها من حواف ومكونات. كما ترى، فإن عملية الإظهار عكس عملية الضبابية blurring. كما ذكرنا من قبل، فإن الضبابية عبارة عن تجميع لشدة البكسلات في المنطقة المجاورة وإيجاد متوسطها وإحلالها محل البكسل الموجودة في مركز هذه المنطقة. أي أنها في النهاية عبارة عن عملية تكامل لشدة البكسلات في هذه المنطقة. لذلك فإننا نتوقع أن تكون عمليات الإظهار sharpening عبارة عن عملية تفاضل (عكس عملية التكامل في مرشحات الضبابية). كما نعلم، فإن التفاضل في الدوال الرقمية عبارة عن الفرق، لذلك فإنه في المنطقة المتجانسة التي لها نفس مستوى الرمادية عند كل البكسلات سيكون تفاضلها بصفر. بينما عند أي حافة أو خط في الصورة يجب أن نتوقع أن التفاضل سيكون كبيرا؛ لأن عند الخط أو الحافة هناك تغير مفاجئ في الشدة.

البرنامج التالي يوضح مثالا على هذا النوع من المرشحات:

```
I=imread('d:\aseel\aseel256.jpg');
J = rgb2gray(I);
J = imresize(J,[256 256]);
imshow(J);title('Original Image');
h = fspecial('laplacian');
I3 = imfilter(J,h);
figure; imshow(I3); title('Laplacian image')
```

لاحظ السواد الزائد في الصورة اللابلاسيان، وهذا متوقع لأننا كما ذكرنا عندما نفاضل نأخذ الفرق، وعندما تكون النقطتان متساويتين فالفرق بينهما يكون صفرا، ويظهر بالطبع باللون الأسود كما نرى إلا عند الحواف أو الخطوط يكون الفرق له قيمة لذلك يظهر بلون رمادي على حسب قيمة هذا الفرق. هنا يظهر سؤال، ماذا سنستفيد من هذه الصورة؟ في الحقيقة إن الصورة الموجودة في الشكل (١٠،١٥) في حد ذاتها تكون غير مفيدة، ولكن في الكثير من التطبيقات عند إضافة هذه الصورة على الصورة الأصلية يحدث تحسين كبير في الصورة وبالذات من ناحية الإظهار sharpening حيث مع إضافة الصورتين يتم التأكيد على الخطوط والحواف.

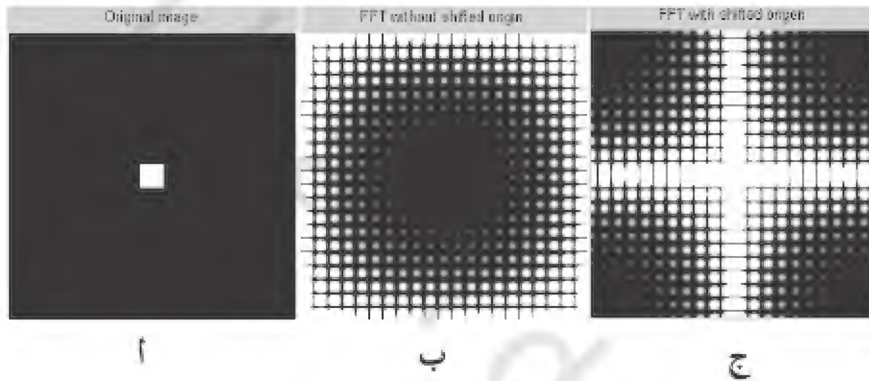


شكل (١٠،١٥). مرشح الإظهار.

(١٠,٥) المعالجة في النطاق الترددي للصورة

إن المعالجة في النطاق الترددي تستلزم الحصول على محول فورير للدالة أو الإشارة، وإذا كانت الإشارة أو الدالة رقمية، فإننا نجرى عليها محول فورير الرقمي.

شكل (١٠.١٦ب و١٠.١٦ج) يبين محول فورير للصورة الموجودة في شكل (١٠.١٦أ)، والبرنامج التالي يبين طريقة الحصول على هذه الصور. لاحظ الفرق بين وجود نقطة الأصل في مركز الصورة ووجودها أعلى يسار الصورة.



شكل (١٠.١٦). محول فورير لصورة أ- الصورة ب- محول فورير بدون إزاحة نقطة الأصل ج- محول فورير بعد إزاحة نقطة الأصل لمركز الصورة.

```
f = zeros(256,256);
f(118:138,118:138) = 1;
imshow(f); title('Original image');
F1 = fft2(f,256,256);
F2 = log(abs(F1));
figure; imshow(F2); title('FFT without shifted origin');
F3 = fftshift(F1);
F4 = log(abs(F3));
figure; imshow(F4); title('FFT with shifted origen');
```

(١٠,٦) تقسيم أو تجزئة الصور

ويعتبر من أهم العمليات على الصور ومن الخطوات الرئيسة في معظم التطبيقات. ويقصد به تقسيم الصورة إلى أجزاء أو عناصر. وتختلف عمليات التقسيم تبعا لنوع الصورة وكذلك المشكلة المراد حلها في التطبيقات المختلفة. فمثلا في الصور الطبية من الممكن أن يكون الهدف من عملية التقسيم هو تحديد مناطق بها أعراض لمعرض معين. وفي الصور الخاصة بالصناعة يكون الهدف تحديد نوع أحد العيوب في منتج. وفي صور الأقمار الصناعية يكون الغرض هو تحديد أماكن بعض الأهداف أو المنشآت. وهكذا، تتعدد الأغراض من عملية تقسيم الصورة وبالتالي تتنوع الطرق وتختلف. ولذلك فإن طرق ونواتج عملية تقسيم الصورة تعتمد على نوعية الصورة.

هنا سنتطرق إلى بعض طرق التقسيم، ولكن يجب أن نذكر أن المراجع المتخصصة [١٤ – ١٨] في مجال معالجة الصور حافلة بالكثير من الطرق الأخرى التي لا مجال لذكرها في هذا الكتاب.

تحديد الخطوط والحواف

يمكن تحديد الخطوط الأفقية عن طريق استخدام نافذة تركز على الخط الأفقي مع المرشح كما سبق وشرحنا في تنعيم الصور. في حالة الخطوط الرأسية والمائلة يمكن تحديدها باستخدام مجموعة من النوافذ كما في شكل (١٠,١٧).

أ

-١	-١	-١
٢	٢	٢
-١	-١	-١

ب

-١	-١	٢
-١	٢	-١
٢	-١	-١

ج

-١	٢	-١
-١	٢	-١
-١	٢	-١

د

٢	-١	-١
-١	٢	-١
-١	-١	٢

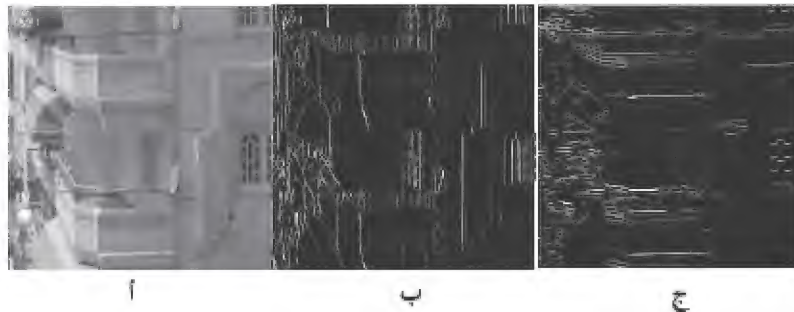
شكل (١٠,١٧). مجموعة من النوافذ لأظهار الخطوط أ– الأفقية ب– المائلة +٤٥° ج– الرأسية د– المائلة –٤٥°.

البرنامج التالي يوضح كيفية تحديد الخطوط الأفقية والرأسية في صورة مبنى عن طريق استخدام النوافذ الموضحة في شكل (١٠،١٧أ) وشكل (١٠،١٧ج). ناتج تنفيذ البرنامج موضح في شكل (١٠،١٨).

```
w1 = [-1 -1 -1;2 2 2;-1 -1 -1];
w2 = [-1 2 -1;-1 2 -1;-1 2 -1];
g1 = imfilter(I,w1);
g2 = imfilter(I,w2);
figure, imshow(g1)
figure, imshow(g2)
```

تحديد الحواف detection Edge

يمكن استخدام الدالة edge الموجودة في الماتلاب مباشرة لتحديد الحواف، حيث تعتمد هذه الدالة على إيجاد المشتقة الأولى أو الثانية للصورة في الاتجاه الأفقي أو الرأسي تبعا لنوع الطريقة المستخدمة لإيجاد وتحديد الحواف. كما ذكرنا في المرشحات، فإنه بالتفاضل يمكن إظهار الخطوط والحواف، ومن هنا ظهر الكثير من الخوارزمات التي تستعمل مرشحات مساحية بمعاملات معينة لإظهار هذه الحواف والتأكيد عليها، من هذه الخوارزمات سوبيل Sobel و برويت Prewitt و روبرتس Roberts و كاني Canny واللابلاسيان Laplacian of Gaussian وتحديد التقاطع مع خط الصفر Zero-crossing.



أ ب ج

شكل (١٠،١٨). تحديد الخطوط والحواف أ- الصورة الأصلية ب- الخطوط الأفقية ج- الخطوط الرأسية.

كل هذه الطرق الست موجودة كاختيارات للدالة edge. شكل (١٠،١٩) يبين تطبيق بعض هذه الخوارزمات على صورة باستخدام البرنامج التالي:

```
I=imread('aseel256.jpg');
J = rgb2gray(I);
J = imresize(J,[256 256]);
imshow(J);title('Original Image');
BW1 = edge(J,'sobel');
figure; imshow(BW1);title('Edges by Sobel');
BW2 = edge(J,'prewitt');
figure; imshow(BW2);title('Edges by Prewitt');
BW3 = edge(J,'roberts');
figure; imshow(BW3);title('Edges by Roberts');
BW4 = edge(J,'canny');
figure; imshow(BW4);title('Edges by Canny');
BW5 = edge(J,'log');
figure; imshow(BW5);title('Edges by Laplacian of Gaussian');
```

نلاحظ من الشكل أن الخرج عبارة عن صورة ثنائية، أي أن قيم البكسلات تأخذ إما القيمة صفر وإما القيمة واحد. يمكن استخدام الدالة edge مع طرق أخرى مثل تحديد التقاطع مع خط الصفر. حاول تجربة ذلك مع الاستعانة بمساعدة الماتلاب عن هذه الدالة باستخدام الأمر:

```
>>help edge
```

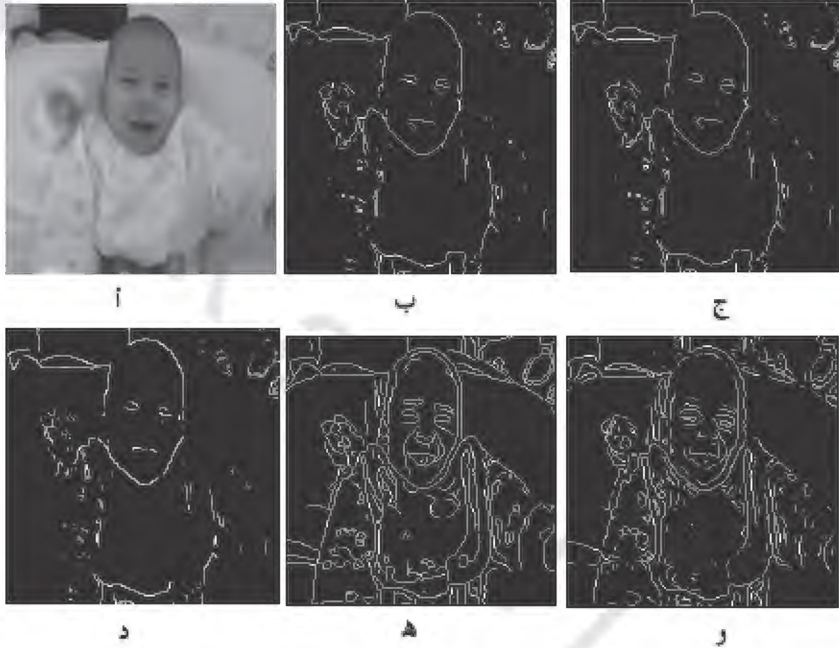
هناك بعض الطرق الأخرى التي تستخدم لتحديد الحواف مثل تحويل هوف Hough الذي يستخدم لتحديد الخطوط والحواف، وفي بعض التطبيقات يستخدم لتحديد الدوائر.

نترك للقارئ تجربة استخدام هذه الطرق مع بعض الصور الموجودة في مكتبة الماتلاب لفهم الفرق بين خرج كل طريقة وتأثير المتغيرات المختلفة مع هذه الطرق.

التقسيم باستخدام حد التشبع Thresholding

عند التعامل مع تقسيم الصورة في أي تطبيق من التطبيقات تكون الخطوة الأولى هي رسم الهستوجرام الخاص بها لمعرفة توزيع مستوى الرمادي في الصورة. ومن الأشياء التي يمكن استنتاجها من الهستوجرام هو تحديد قيمة لحد التشبع، والتي يمكن

عن طريقها تحويل الصورة إلى صورة ثنائية تأخذ القيمة ١ أو اللون الأبيض لكل بكسل تنتمي إلى المنطقة الأولى (منطقة الاهتمام region of interest) وتأخذ القيمة ٠ أو اللون الأسود لكل بكسل تنتمي إلى المنطقة الثانية (الخلفية).



شكل (١٠،١٩). الحصول على حواف وخطوط الصورة أ- الصورة الأصلية ب- الصورة الناتجة بعد استخدام خوارزميات سوبل ج- برويت د- روبرتس هـ - كاني و- لابلاسيان.

تمارين محلولة

١٠- ١ اكتب برنامجا يقرأ الصور الآتية، ويحسب حجمها، ثم يعرض كل منها في شكل منفصل بحيث يكون عنوان الشكل هو اسم الصورة :

١- tire.tif
٢- moon.tif
٣- cameraman.tif

٤- peppers.png

```
%Solution of Excersis 10-1
%To read an image and calculate its size
I1=imread('tire.tif');
[r1,c1,d1]=size(I1)
figure,imshow(I1),title('tire.tif')
```



شكل (١٠،٢٠). تمرين١٠-١ الصورة الأولى.

```
I2=imread('moon.tif');
[r2,c2,d2]=size(I2)
figure,imshow(I2),title('moon.tif')
```



شكل (١٠،٢١). تمرين ١٠-١ الصورة الثانية.

```
I3=imread('cameraman.tif');
[r3,c3,d3]=size(I3)
figure,imshow(I3),title('cameraman.tif')
```



شكل (١٠،٢٢). تمرين ١٠-١ الصورة الثالثة.

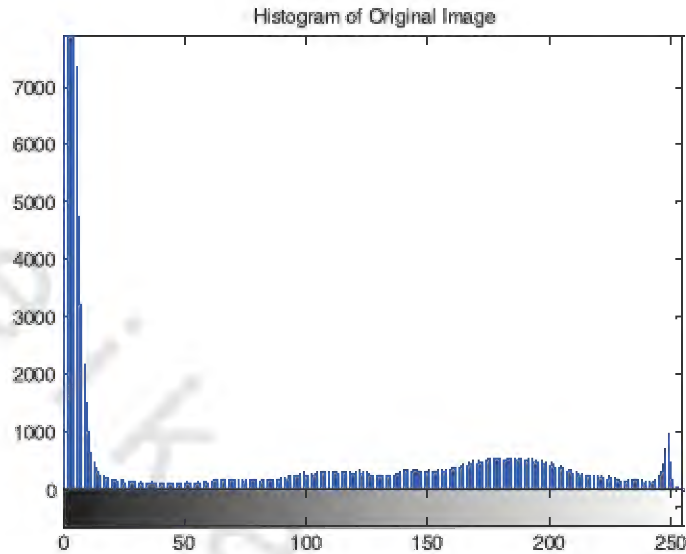
```
I4=imread('peppers.png');
[r4,c4,d4]=size(I4)
figure,imshow(I4),title('peppers.png')
```



شكل (١٠,٢٣). تمرين ١٠-١ الصورة الرابعة.

١٠- ٢ اكتب برنامجا لعرض هستوجرام الصورة moon.tif ثم يقوم بعمل تعديل للهستوجرام.

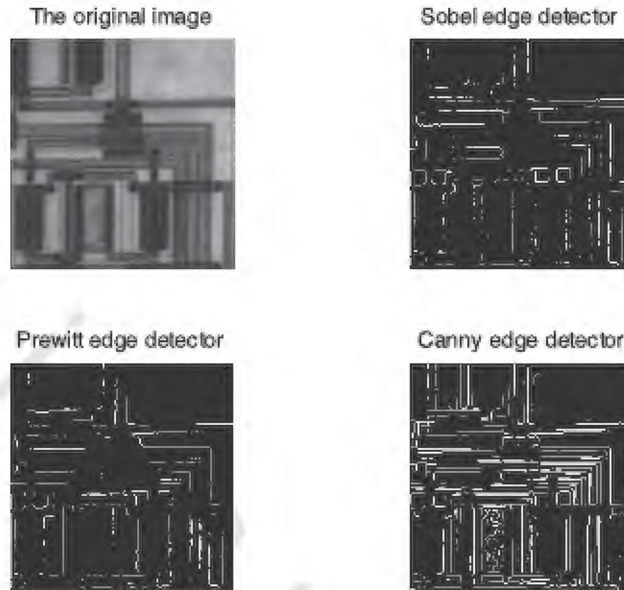
```
% Solution of exercise 10-2
I=imread('moon.tif');
figure,imhist(I),title('Histogram of Original Image')
I_eq=imadjust(I,[]);
```



شكل (١٠,٢٤). تمرين ١٠-٢.

١٠- ٣ اكتب برنامجا لتحديد الحواف باستخدام الطرق المختلفة للصورة:

```
%Solution of exercise 10-3
C=imread('circuit.tif');
E1=edge(C,'sobel');
E2=edge(C,'prewitt');
E3=edge(C,'canny');
figure,
subplot(2,2,1),imshow(C),title('The original image')
subplot(2,2,2),imshow(E1),title('Sobel edge detector')
subplot(2,2,3),imshow(E2),title('Prewitt edge detector')
subplot(2,2,4),imshow(E3),title('Canny edge detector')
```



شكل (١٠,٢٥). تمرين١٠-٣.

١٠- ٤ استخدم الدالة im2bw لتحويل الصورة rice.png إلى صورة ثنائية القيمة باستخدام قيم مختلفة لحد التشبع.

```
%Solution of exercise 10-4
R=imread('rice.png');
figure,
BW1=im2bw(R,0.25);
BW2=im2bw(R,0.5);
BW3=im2bw(R,0.75);
subplot(2,2,1), imshow(R),title('The original image')
subplot(2,2,2), imshow(BW1),title('Thresold = 0.25')
subplot(2,2,3), imshow(BW2),title('Thresold = 0.5')
subplot(2,2,4), imshow(BW3),title('Thresold = 0.75')
```





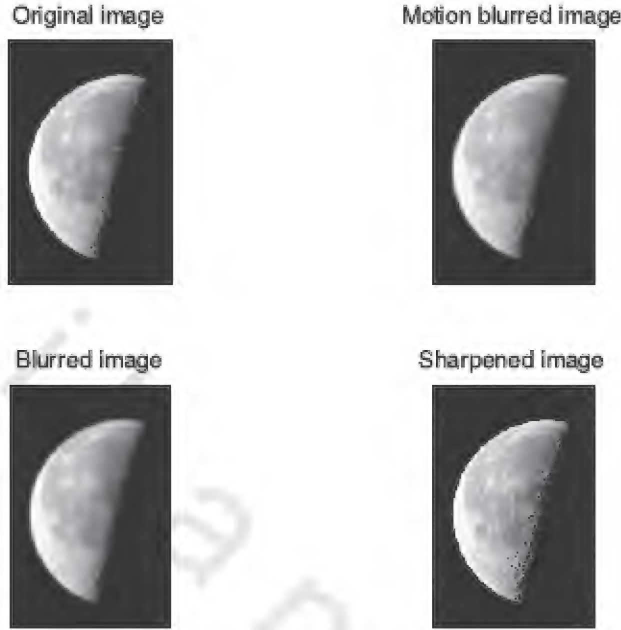




شكل (١٠,٢٦). تمرين١٠-٤.

١٠- ٥ استخدم الدالة fspecial لعمل أنواع مختلفة من التنعيم للصورة moon.tif :

```
%Solution of exercise 10-5
I=imread('moon.tif');
subplot(2,2,1);imshow(I);title('Original Image');
H=fspecial('motion',10,25);
MotionBlur=imfilter(I,H,'replicate');
subplot(2,2,2);imshow(MotionBlur);title('Motion Blurred
Image');
H=fspecial('disk',5);
blurred=imfilter(I,H,'replicate');
subplot(2,2,3);imshow(blurred);title('Blurred Image');
H=fspecial('unsharp');
sharpened=imfilter(I,H,'replicate');
subplot(2,2,4);imshow(sharpened);title('Sharpened
Image');
```



شكل (١٠،٢٧). تمرين١٠-٥.

سنكتفي بهذا القدر كمقدمة عن استخدام الماتلاب في المعالجة الرقمية للصور، فكما أوضحنا قبلا أن استخدام الماتلاب في معالجة الصور يجب أن يكون في كتاب خاص به. ولكننا فقط أردنا أن نقدم للقارئ مقدمة عن أساسيات استخدام الماتلاب في معالجة الصور الرقمية. وعلى القارئ المهتم بهذا التخصص اللجوء إلى المراجع المتخصصة في هذا المجال وما أكثرها، ومنها ما ورد ذكره في الجزء الخاص بالمراجع [١٤-٢٠].

الفصل الحادي عشر

معالجة الإشارات

(١١,١) مقدمة

بعد اكتساب البيانات وإدخالها إلى الحاسب يمكن إجراء الكثير من المعالجات عليها. من هذه المعالجات، مثلا ترشيح هذه الإشارة للتخلص من الترددات غير المرغوبة، أو رسم المحتويات الترددية لهذه الإشارة من خلال إجراء محول فورير عليها، كما يمكن رسم طيف القدرة لهذه الإشارة، وتوليد أشكال مختلفة للإشارات. في هذا الفصل نقدم مقدمة عن كيفية معالجة الإشارات باستخدام برنامج الماتلاب. في البداية نتعرض لطرق توليد الأشكال الموجية وإضافة ضوضاء إليها، وبعد ذلك ننتقل إلى شرح تصميم المرشحات ثم تحليلها من خلال شاشات التفاعل مع المستخدم.

سيتم ذلك من خلال مجموعة من الأوامر والشاشات التفاعلية المجمعة كلها في صندوق من صناديق أدوات ماتلاب، والذي يسمى signal processing tool box. جميع أوامر معالجة الإشارة تتعامل مع بيانات من النوع المتضاعف الدقة double precision. إذا تم إدخال أي نوع آخر من البيانات، فإن ماتلاب ربما يعطي رسالة خطأ تدل على ذلك. معظم دوال أو أوامر معالجة الإشارة عبارة عن خوارزمات مبنية في ملفات من

النوع M files. تتعامل دوال معالجة الإشارة مع كل من الإشارات والأنظمة سواء في ذلك الإشارات والأنظمة التماثلية وكذلك الإشارات والأنظمة الرقمية. الأنظمة – وبالذات المرشحات – التي سيتم التعامل معها هنا هي الأنظمة الخطية غير المتغيرة زمنيا Linear Time Invariant, LTI الواسعة الانتشار والاستخدام.

(١١.٢) توليد الأشكال الموجية

في العادة يتطلب توليد أي شكل موجي الحصول على متجه زمني له بداية ونهاية وله خطوة. في العادة يتم ذلك بأمر كالتالي :

```
>> t = (0:0.001:1)';
```

الذي يولد متجه زمني عبارة عن صف من ١٠٠١ عنصر يبدأ من القيمة صفر وينتهي بالقيمة واحد وخطوته مقدارها ٠.٠٠١ أو واحد ميللي ثانية. هناك العملية (') والتي تعنى العكس أو transpose والتي ينتج عنها تحويل هذه المصفوفة من مصفوفة صف واحد إلى مصفوفة عمود واحد. بالطبع كما علمنا من قبل، فإن الفاصلة المنقوطة (؛) في نهاية الأمر تعنى أن ماتلاب سيحسب هذا المتجه، ولكنه لن يعرضه على شاشة الحاسب في مجال العمل.

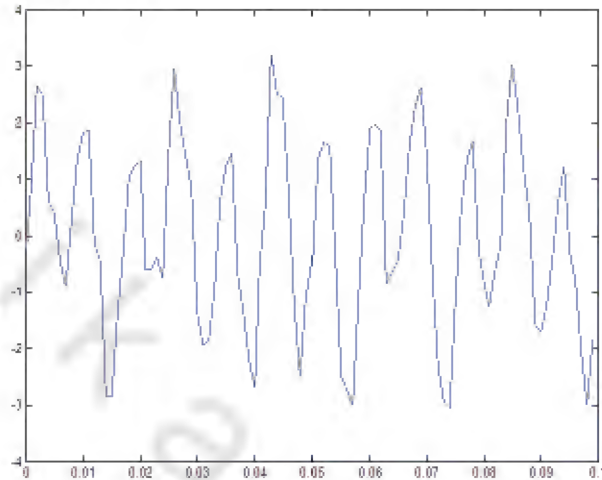
الآن يمكن الحصول على العديد من الموجات الجيبية باستخدام هذا المتجه الزمني، وكمثال على ذلك الموجة y التالية التي تتكون من مجموع موجة جيبية بتردد ٥٠ هرتزاً مع أخرى بتردد ١٢٠ هرتز ومقدارها ضعف مقدار الأولى كما يلي :

```
>> y = sin(2*pi*50*t) + 2*sin(2*pi*120*t);
```

يمكنك إضافة ضوضاء للإشارة y ورسم أول ١٠٠ نقطة منها كما يلي، وكما هو مبين في شكل (١١.١) :

```
>> t=(0:0.001:1)';
>> y = sin(2*pi*50*t) + 2*sin(2*pi*120*t);
```

```
>> randn('state',0);
>> yn = y + 0.5*randn(size(t));
>> plot(t(1:100),yn(1:100))
```



شكل (١١،١). مجموع موجتين جيبيتين مع ضوضاء.

حيث الدالة randn تولد رقماً عشوائياً بتوزيع طبيعي normal distribution ومتوسط mean يساوي صفراً.

يمكن توليد دالة نبضة الوحدة impulse باستخدام أمر كالتالي :

```
>> y = [1; zeros(99,1)];
```

وهي عبارة عن مصفوفة عمود يتكون من ١ في أول صف ويليه ٩٩ صفراً.

يمكن توليد الخطوة الواحدة أو وحدة الخطوة unit step باستخدام كالتالي :

```
>> y = ones(100,1);
```

وهي عبارة عن عمود من ١٠٠ صف كل منها به القيمة واحد.

يمكن توليد دالة الصعود (أو الانحدار) كما يلي :

```
>> t = (0:0.001:1)';
>> y = t;
```

دالة التربيع يمكن توليدها كالتالي :

```
>> y = t.^2;
```

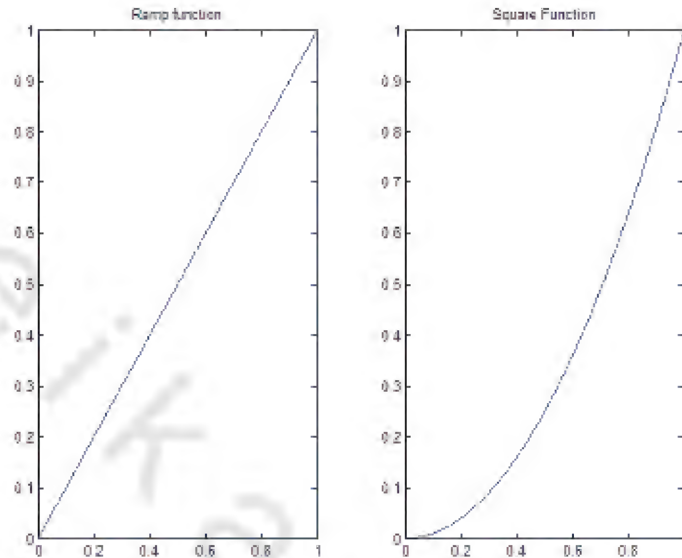
شكل (١١,٢) يبين كل من دالة الصعود ودالة التربيع .

الإشارة المكونة من عمود واحد تسمى إشارة ذات قناة واحدة single channel signal مثل الإشارة التالية التي تتكون من قناة واحدة (عمود واحد) كل صف فيه (أو كل عينة) تساوي واحداً :

```
>> a=[1 ones(1,3)]'
a =
    1
    1
    1
    1
```

الإشارة المكونة من أكثر من قناة multichannel تتكون من أكثر من عمود، كل عمود يمثل قناة، وكل صف من هذه القنوات يمثل (عينة). فمثلا يمكن توليد إشارة ذات ثلاث قنوات بدلالة الإشارة a السابقة كما يلي :

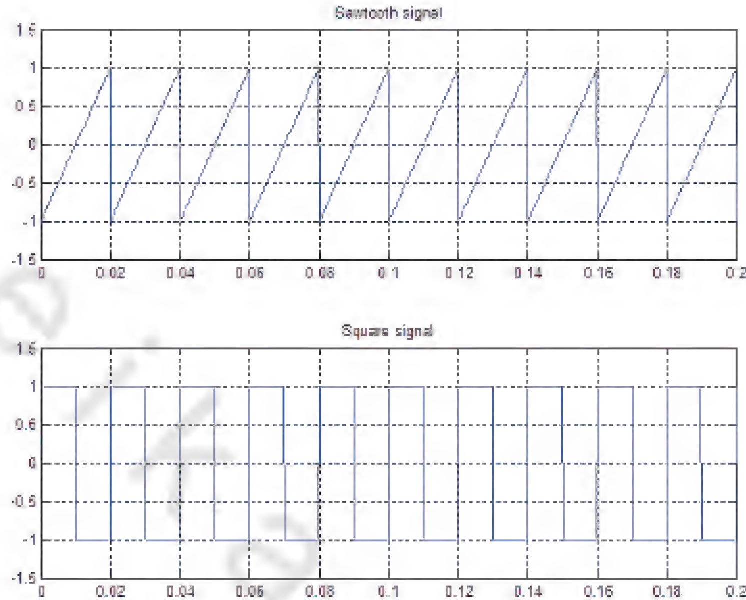
```
>> b=[a 2*a 3*a]
b =
    1    2    3
    1    2    3
    1    2    3
    1    2    3
```



شكل (١١,٢). دالتا الصعود والتربيع.

يحتوي ماتلاب على بعض الإشارات الدورية ، مثل إشارة سن المنشار sawtooth والإشارة المربعة square بخلاف الدالتين sin و cos. البرنامج التالي يرسم كل من هاتين الدالتين كما في شكل (١١,٣).

```
fs = 10000;
t = 0:1/fs:1.5;
x = sawtooth(2*pi*50*t);
subplot(2,1,1)
plot(t,x), axis([0 0.2 -1.5 1.5])
grid
title('Sawtooth signal')
x = square(2*pi*50*t);
subplot(2,1,2)
plot(t,x), axis([0 0.2 -1.5 1.5])
title('Square signal')
grid
```



شكل (١١.٣). موجة سن المنشار والموجة المربعة بدوال جاهزة في ماتلاب.

يحتوي ماتلاب أيضا على العديد من الدوال غير الدورية ، مثل الدالة gauspulse التي تولد نبضة جاوسية بتردد راديو عند زمن t، وبتردد مركزي fc هرتز وعرض مجال bw، والصورة العامة لهذه الدالة هي :

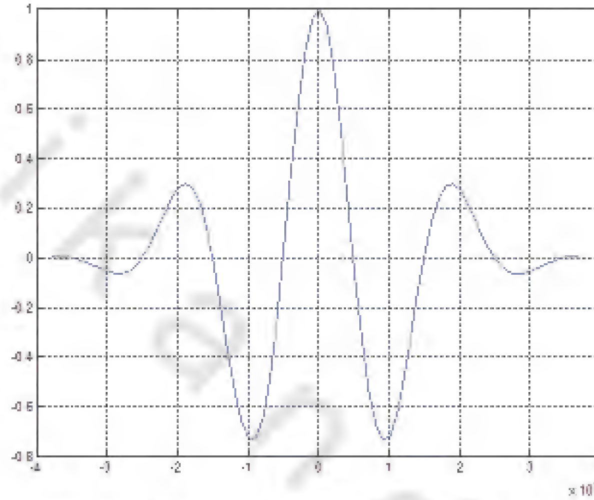
yi = gauspuls(t,fc,bw)

الأوامر التالية تعطي نبضة جاوسية بتردد ٥٠kHz وعرض مجال ٦٠٪ ومعدل عيناتها يساوي واحداً ميجاهرتز، وأما المحيط الخارجي لهذه الدالة فيتناقص بمقدار ٤٠dB تحت قمة النبضة.

شكل (١١.٤) يبين هذه الدالة المرسومة بناء على الأوامر التالية :

```
tc = gauspuls('cutoff',50e3,0.6,[],-40);
t = -tc : 1e-6 : tc;
yi = gauspuls(t,50e3,0.6);
plot(t,yi)
```

هناك بعض الدوال الأخرى الدورية وغير الدورية، والتي منها الدوال التالية: chirp, cos, diric, gauspuls, rectpuls, sawtooth, sin, sinc, square, tripuls. وسنترك للقارئ المهتم بأي واحدة من هذه الدوال أن يقرأ المساعدة help الموجودة في ماتلاب.



شكل (١١,٤). النبضة الجاوسية كدالة غير دورية جاهزة في ماتلاب.

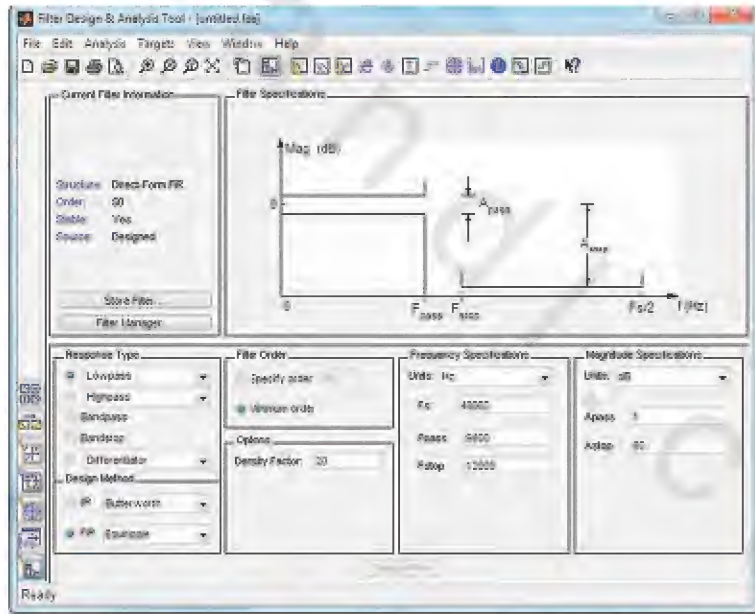
(١١,٣) تصميم وتحليل المرشحات من خلال شاشات التفاعل مع المستخدم

تصميم وتحليل المرشحات يعتبر من أهم أجزاء أو أبواب أي مؤلف أو أي كتاب خاص بمعالجة الإشارات. برنامج ماتلاب يقدم شاشات تفاعل مع المستخدم يمكنه من خلالها تصميم وتحليل أي مرشح من أي نوع يريده وبسرعة فائقة يمكنه أن يرى الاستجابة الترددية لهذا ومعاملاته.

يمكن الدخول على شاشات التفاعل هذه من خلال الأمر fdatool والتي هي اختصار للمعنى، أدوات تصميم وتحليل المرشح filter design and analysis tool. لذلك من مجال العمل work space في ماتلاب سنكتب الأمر fdatool، حيث ستظهر لك الشاشة الأساسية لتصميم المرشح الموجودة في شكل (١١,٥).

يمكن تقسيم شاشة الفعالية في شكل (١١,٥) إلى عدة أقسام، الجزء العلوي من اليمين يبين الاستجابة الترددية للمرشح، والتي تعطي العلاقة بين مقدار خرج المرشح منسوبا إلى دخله بالديسبل مع التردد بالهرتز مبينا الترددات الحرجة، مثل تردد القطع cut of freq. وتردد مجال الوقف band stop freq. وتردد مجال المرور pass band freq..

في الجزء العلوي من اليسار ترى مربعا يحتوي ملخصا لمعاملات هذا المرشح. لاحظ أن المرشحات التي يتم التعامل معها هنا هي مرشحات رقمية digital filters وليست تماثلية. في البداية يفترض ماتلاب قيما تلقائية لمرشح افتراضي يتم افتراضه في بداية التعامل، ويمكن رؤية معاملات هذا المرشح في أول شاشة يتم فتحها.



شكل (١١,٥). الشاشات الفعالية مع المستخدم لتصميم وتحليل المرشحات.

في الجزء الأسفل من الشاشة الموضحة في شكل (١١,٥) يوجد أكثر من مربع يمكن من خلالها اختيار التصميم للمرشح المطلوب من حيث نوعه هل هو مرشح

للترددات المنخفضة أم للترددات المرتفعة أم لمجال معين من الترددات؟ كما يمكن اختيار ترددات القطع المختلفة ومعدل العينات وغير ذلك من المعاملات كما سنرى.

١- مثلا، من مربع اختيار نوع الاستجابة Response Type، اختر Bandpass أي منفذ لمجال من الترددات.

٢- من خلال مربع اختيار طريقة التصميم Design Method، اختر IIR، ثم اختر Butterworth من قائمة الاختيار.

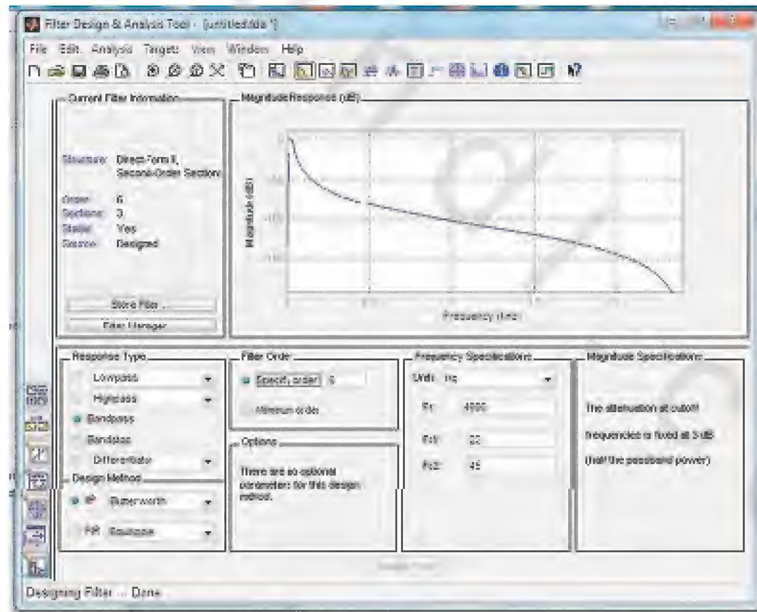
٣- اختر درجة المرشح واجعلها تساوي ٦ من العنوان Specify order.

٤- اضبط ترددات المرشح لتكون كما يلي: تردد العينات sampling frequency يساوي Fs=4800Hz، تردد القطع الأول (عندها يتناقص المقدار ٣ ديسبل قبل مجال المرور) Fc1=22Hz، تردد القطع الثاني (عندها يتناقص المقدار ٣ ديسبل بعد مجال المرور) Fc2=45Hz.

٥- بعد الانتهاء من تحديد معاملات المرشح، اضغط على زر تصميم المرشح Design Filter، حيث يقوم ماتلاب بحساب المرشح ورسم مقدار الاستجابة الترددية للمرشح. لاحظ أن زر تصميم المرشح يصبح غير فعال بعد النقر عليه، ولكن عند تغيير أي واحد من معاملات المرشح، فإن هذا الزر ينشط مرة ثانية بحيث يقوم بمجرد الضغط عليه مرة ثانية بتغيير الاستجابة على حسب المعاملات الجديدة. شكل (١١,٦) يبين شاشة تصميم المرشح عقب انتهاء مراحل اختيار معاملات التصميم المختلفة.

٦- في نهاية مرحلة التصميم يمكنك تخزين هذا المرشح حسب آخر معاملات تم التعامل معها بالنقر على الزر Store Filter في أسفل المربع العلوي يسارا حيث سيفتح لك مربعاً حواريًا تختار من خلاله اسم لهذا المرشح.

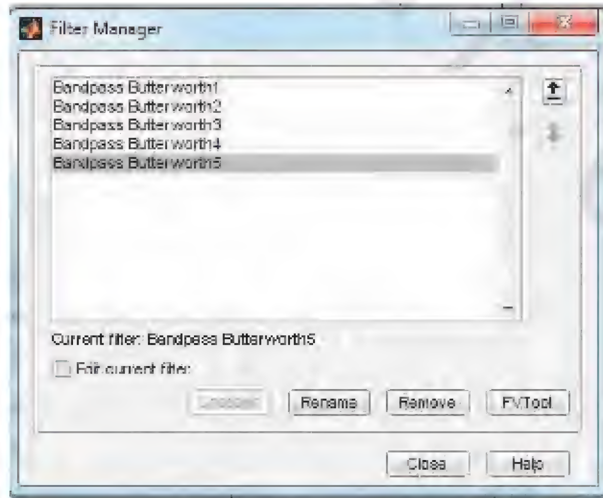
٧- الشاشة الرئيسة لتصميم المرشح تحتوي في أعلاها على شريط أيقونات يمكنك التجربة مع كل منها لتعرض معاملات واستجابة المرشح بطرق مختلفة. هناك مثلا أيقونة بالنقر عليها يعرض أمامك أصفاراً وأقطاب المرشح في المستوى Z. يمكنك رؤية الاستجابة الطورية للمرشح وأيضا تأخير المجموعة Group delay، كما يمكنك رؤية استجابة الخطوة Step response واستجابة الصدمة implse response. هناك أيضا في أعلى هذه الشاشة شريط للقوائم، والتي منها شريط التحرير Edit الذي يمكنك من خلاله نسخ الاستجابات المختلفة للمرشح وتغيير بداية ونهاية وخطوة رسم كل واحد من محاور الرسم. هذه الشرائط تستحق أن تبذل بعض الوقت في تجربتها لترى الكثير من الإمكانيات التي يمكنك الحصول عليها والتي تقع خارج نطاق الكتاب.



شكل (١١,٦). شاشة تصميم المرشح بعد نهاية خطوات التصميم.

٨- يمكنك تصميم أكثر من مرشح وعرضها في نفس الشاشة لترى الاستجابات المختلفة لكل منها بغرض المقارنة. لعمل ذلك سنضيف تصميم ٤مرشحات أخرى بترددات قطع مختلفة عن المرشح السابق، ولكن مع الاحتفاظ بنفس تردد العينة Sampling frequency. لذلك سنرجع مرة ثانية إلى شاشة تصميم المرشح الأساسية السابقة، ونترك جميع المعاملات كما هي سواء أننا سنغير Fc1=45 و Fc2=89 ونضغط على زر التصميم ثم على زر التخزين لنخزن المرشح الجديد بالاسم Butterworth filter2. ثم نغير FC1 و Fc2 إلى Fc1=89 وFc2=187 ثم نضغط زر التصميم وزرار التخزين ونخزن المرشح بالاسم Butterworth filter3 وهكذا نكرر هذه الخطوات حتى أي عدد من المرشحات نريده.

٩- في شاشة تصميم المرشحات الأساسية؛ اضغط على الزر File Manager في المربع الأيسر في أعلى الشاشة حيث ستظهر أمامك نافذة جديدة تحتوي أسماء جميع المرشحات التي تم تصميمها في الخطوة ٨. شكل (١١,٧) يبين هذه الشاشة التي تحتوي على خمسة مرشحات تم تصميمها .



شكل (١١,٧). متحكم المرشحات.

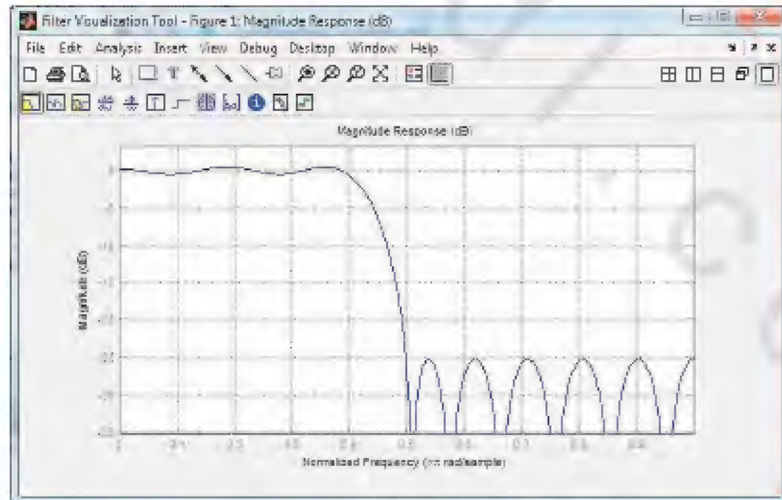
تحتوي هذه الشاشة أيضا على زر Remove يمكن من خلاله حذف أي مرشح من هذه المجموعة. كما تحتوي على الزر Rename الذي يمكن استخدامه لتغيير اسم أي مرشح من هذه المرشحات بعد النقر عليه.

في الجزء السابق أوضحنا كيفية تصميم وتحليل المرشحات من خلال شاشات التفاعل مع المستخدم باستخدام الأمر fdatool. وقبل أن نختم حديثنا في موضوع معالجة الإشارات باستخدام الماتلاب، نستعرض سريعا الدالة fvtool والدالة sptool.

الدالة fvtool(b,a) والتي هي اختصار للمعنى، أدوات رؤية المرشح Filter visualization tool تفتح شاشة تفاعلية وتحسب مقدار استجابة المرشح الرقمي المعرف بقسمة b على a.

شكل (١١،٨) يوضح تنفيذ الأمر التالي:

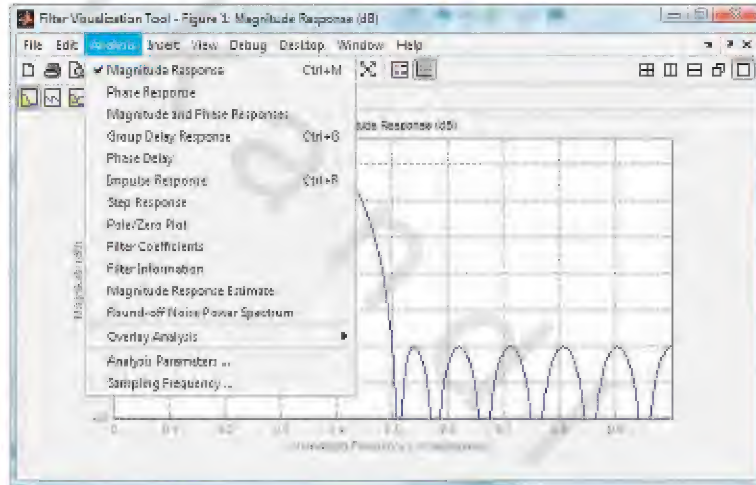
```
>>b1 = firpm(20,[0 0.4 0.5 1],[1 1 0 0]);
>> fvtool(b1,1);
```



شكل (١١،٨). الشاشة التفعالية للدالة fvtool.

عن طريق هذه الدالة يمكن عرض التأخير group delay، الطور phase response، استجابة الصدمة impulse response، استجابة الخطوة step response، رسم الأقطاب والأصفر pole/zero plot، معاملات المرشح coefficients.

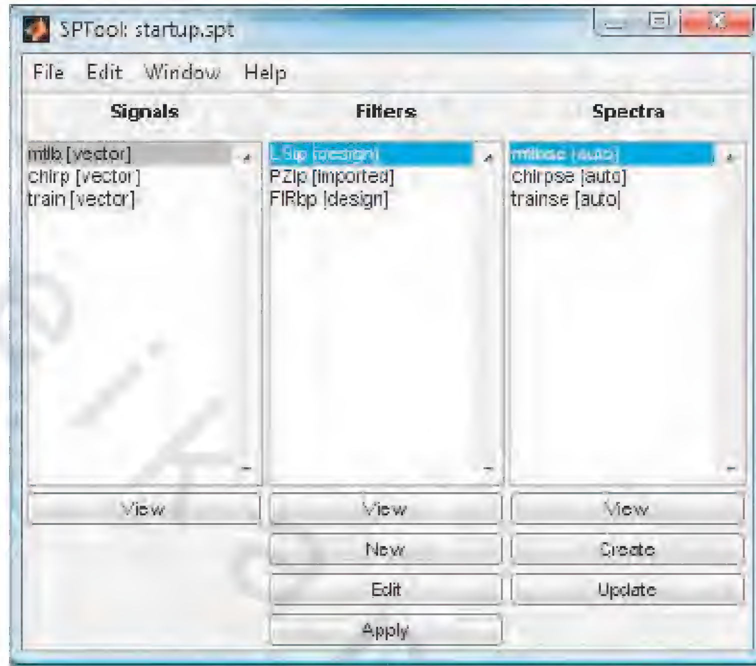
قائمة Analysis الموجودة في شريط القوائم تعرض كل الإمكانات المتاحة للمستخدم كما هو موضح في شكل (١١،٩). كذلك يوجد شريط للأدوات يمكن من خلاله تجربة هذه الإمكانيات.



شكل (١١،٩). قائمة Analysis.

والدالة sptool والتي هي اختصار للمعنى، أدوات معالجة الإشارات Signal processing tool تفتح شاشة تفاعلية مع المستخدم تمكنه من أربعة أدوات مختلفة. أحداها خاصة بتصفح الإشارات signal browser، والثانية بتصميم المرشحات filter design، والثالثة fdatool، أما الأخيرة فهي تختص بمعاينة ومشاهدة الطيف spectrum viewer. شكل (١١،١٠) يعرض هذه الشاشة كناتج تنفيذ الأمر التالي:

```
>> sptool
```



شكل (١١,١٠). الشاشة التفعالية للدالة sptool.

في نهاية هذا الفصل ننصح القارئ أن يستكشف الدالة fvtool والدالة sptool عن طريق تجربة الأدوات المتاحة لكل منها. للمزيد من الأمثلة والتطبيقات ننصح القارئ بمراجعة بعض فصول المرجع [٢٢].

المراجع

1. MATLAB: Getting Started Guide, The Math Works, 2011.
2. MATLAB, The Language of Technical Computing: Using MATLAB Graphics Version 7, The Math Works, 2004.
3. MATLAB, The Language of Technical Computing: Getting Started with MATLAB Version 6, The Math Works, 2001.
4. MATLAB Programming, David Kuncicky, Pearson, 2003.
5. Essential Matlab for Engineers and Scientists, *Brian H. Hahn and Daniel T. Valentine, 3rd* Edition, Butterwort-Heinemann, 2007.
6. Matlab Primer, Kermit Sigmon and Timothy A. Davis, 7th Edition, Chapman & Hall/CRC, 2005.
7. Practical MATLAB Basics for Engineers, Misza Kalechman, CRC Press, 2009.
8. Practical MATLAB Applications for Engineers, Misza Kalechman, CRC Press, 2009.
9. Creating Graphical User Interface, The Math works, Version 7, 2004.
10. Graphics and GUIs with MATLAB, Patrick Marchand and O. Thomas Holland, 3rd Edition, Chapman & Hall/CRC, 2003.
11. MATLAB: Advanced GUI Development, Scott T. Smith, Dog Ear Publishing, 2006.
12. Introduction to Simulink with Engineering Applications, Steven T. Karris, 2nd Edition, Orchard Publications, 2008.
13. Signals and Systems with MATLAB Computing and Simulink Modeling, Steven T. Karris, 4th Edition, Orchard Publications, 2008.
14. Digital Image Processing, Rafael C. Gonzalez and Richard E. Woods, 3rd Edition, Prentice Hall, 2008.
15. Digital Image Processing Using MATLAB, Rafael C. Gonzalez, Richard E. Woods, and Steven L. Eddins, McGraw-Hill Education, 2004.

16. Image Processing, Analysis, and Machine Vision, Milan Sonka, Vaclav Hlavac, and Roger Boyle, Chapman and Hall Computing, London, 1993
17. Feature Extraction and Image Processing, Mark Nixon and Alberto Aguado, Elsevier Science, 2002.
18. Image Processing with MATLAB: Applications in Medicine and Biology, Omer Demirkaya, Musa Hakan Asyali and Prasanna K. Sahoo, CRC Press, 2009.
19. Biomedical Signal and Image Processing, Kayvan Najarian and Robert Splinter, CRC Press, 2006.
20. Biosignal and Biomedical Image Processing: MATLAB-Based Applications, John L. Semmlow, Marcel Dekker, 2004.
21. Adaptive Filter Theory, Simon Haykin, 2nd Edition, Prentice-Hall, 1991.
22. Digital Signal Processing Using Matlab, André Quinquis, Wiley-ISTE, 2008.
23. Introduction to Signal Processing, Sophocles J. Orfanidis, Prentice Hall, 1996.
24. Signal Processing Systems: Theory and Design, N. Kalouptsidis, John Wiley &Sons, 1997.
25. Signal Processing & Linear Systems, B.P. Lathi, Oxford University Press, 1998.
26. Linear Systems and Signals, B.P.Lathi, Oxford University Press, 2005.
27. Digital Signal Processing with MATLAB, Vinay K. Ingle and John G. Proakis, Brooks/Cole, 2000.
28. Numerical Analysis and Graphic Visualization with MATLAB, Shoichiro Nakamura, 2nd Edition, Prentice Hall, 2002.
29. Applied Numerical Methods Using MATLAB, Won Y. Yang, Wenwu Cao, Tae-Sang Chung, and John Morris, Wiley-Interscience. 2005.

كشاف الموضوعات

أ

ب

ت

ح

د

ر

س

ط

ع